VisiCalc

Douglas Hergert

SYBEX

EDIZIONE ITALIANA

Douglas Hergert

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

SOMMARIO

APPENDICI

RINGRAZIAMENTI

I miei più sinceri rigraziamenti vanno a Salley Oberlin ed Elaine Foster per il sostegno tecnico, editoriale e morale che mi hanno prestato durante la preparazione di questo libro. Desidero anche ringraziare i seguenti membri del reparto produzione della Sybex per la loro attenta opera: Valerie Brewster, Margaret Cusick, Michael Howard, Sharon Leong, Ingrid Owen, Donna Scanlon, Hilda van Genderen e Cheryl Wilcox.

INTRODUZIONE

I programmi di "foglio elettronico" sono rapidamente divenuti una delle principali componenti del software su personal computer. Si tratta di programmi che consentono di elaborare simultaneamente centinaia di numeri, trovare schemi, trarre conclusioni ed esplorare alternative suggerite da dati numerici di qualunque tipo. Le loro possibilità sono utili tanto per le limitate esigenze matematiche della vita quotidiana che per le più imponenti operazioni contabili e decisionali dell'ambiente lavorativo. Ma l'aspetto forse più interessante è che l'uso di questi programmi non richiede che le conoscenze tecniche e le capacità matematiche necessarie per usare una piccola calcolatrice.

Il programma VisiCalc®, messo a punto dalla Software Arts, Inc., è senz'altro uno dei migliori e più diffusi del suo genere. E' disponibile per la maggior parte dei principali microcomputer attualmente sul mercato e ha le due qualità più importanti: semplicità di utilizzazione; versatilità e flessibilità sufficienti per la soluzione di un'ampia gamma di problemi tecnici e non.

Aspetto ancor più importante, i creatori del VisiCalc hanno fatto in modo che il loro programma fosse in grado di scambiare, in modo semplice e affidabile, dati con altri strumenti di software residenti nello stesso ambiente di microcomputer. Tale cordialità è rara nel mondo del software commerciale: sono troppi i programmi volutamente progettati così da essere inavvicinabili, non cooperanti e monolitici rispetto ad altri programmi concorrenti.

Mastering VisiCalc è stato scritto per utenti di vari livelli:

- Per chi si avvicina al VisiCalc per la prima volta, questo libro costituisce una guida nei primi passi dell'apprendimento dell'uso del programma per necessità individuali. Viene spiegato come inserire i dati e come sfruttare tutte le principali possibilità del VisiCalc.

- Per chi ha già cominciato a usare il VisiCalc e ha dubbi su alcune delle sue possibilità più avanzate, questo è un libro utile: vi si troveranno spiegazioni dettagliate delle sottigliezze delle funzioni più difficili - come @LOOKUP, @NVP e le funzioni logiche - e vi vengono insegnate le tecniche per l'utilizzazione ottimale del programma.

- Infine, per chi è pronto ad andare oltre e cominciare a utilizzare il VisiCalc con altri programmi da lui scritti, questo libro offre un'introduzione completa

all'uso dei file DIF per lo scambio di dati fra il VisiCalc e il BASIC. Padroneggiare questa caratteristica può rappresentare una sfida, ma chi lo farà potrà avvalersi di un controllo molto maggiore sulle attività di elaborazione effettuate tramite il computer.

Capitolo 1 - *Il programma VisiCalc e il computer* - Introduce l'argomento del libro. Presenta diversi esempi sulle possibilità del VisiCalc e dà una rapida visione d'insieme delle caratteristiche del programma.

Capitolo 2 - *Un'introduzione al VisiCalc* - descrive in modo completo e dettagliato i principali comandi del VisiCalc: mostra come inserire i dati, scrivere le formule e applicarle a righe e a colonne di dati.

Capitolo 3 - *Le funzioni del VisiCalc, Parte I* - spiega l'uso delle funzioni più comuni, come @SUM e @AVERAGE, e dà modo di studiare l'azione delle funzioni matematiche più avanzate.

Capitolo 4 - *Le funzioni del VisiCalc, Parte II* - guida nell'analisi della funzione @LOOKUP e di quelle logiche, che talvolta appaiono misteriose al principiante. Poichè alcune versioni del VisiCalc non comprendono le funzioni logiche, questo capitolo suggerisce diversi metodi alternativi per simularle.

Capitolo 5 -*Costruzione dei fogli elettronici del VisiCalc* - è un capitolo fondamentale per tutti i lettori. Mostra come creare fogli di formato generale da utilizzare più di una volta. Chiarisce anche alcuni concetti che sono potenziali fonti di confusione riguardanti il modo in cui il VisiCalc effettua i calcoli sul foglio.

Gli ultimi tre capitoli del libro introducono i file DIF, che rappresentano un metodo per passare dati dal programma VisiCalc al BASIC. Chi non ha mai programmato in BASIC e i principianti in materia dovrebbero leggere l'Appendice B - *Uno sguardo al BASIC* - prima di cominciare questi capitoli.

Capitolo 6 - *I file DIF, Parte I* - descrive e illustra la struttura del *Data Interchange Format* (DIF - formato per l'interscambio dei dati). Presenta anche un programma in BASIC che mostra qualunque file DIF sullo schermo del computer. Questo programma, e quelli che seguono nei Capitoli 7 e 8, richiedono alcuni strumenti di gestione dei file le cui istruzioni variano a seconda delle versioni del BASIC. L'Appendice A mostra queste istruzioni per i BASIC di tre personal computer fra i più diffusi: Apple, TRS-80 e IBM.

Capitolo 7 - *I file DIF, Parte II* - tratta situazioni che possono richiedere le possibilità sia del VisiCalc che del BASIC. Il capitolo mostra come un programma in BASIC può accedere ai dati di un foglio elettronico del VisiCalc.

Capitolo 8 - *I file DIF, Parte III* - mostra il passaggio finale nello scambio di dati fra VisiCalc e BASIC. Il capitolo presenta uno strumento che si può dimostrare prezioso in molte situazioni in cui si devono elaborare dati: un programma in BASIC per l'ordinamento dei dati di un foglio elettronico del VisiCalc.

In conclusione, *Conoscere il Visicalc* ha un duplice scopo: insegnare a usare il programma VisiCalc e aiutare a esplorare l'utilizzazione del VisiCalc su un personal computer in combinazione con altri programmi.

CAPITOLO 1

IL PROGRAMMA VISICALC E IL COMPUTER

INTRODUZIONE

Questo primo capitolo è una presentazione di quanto il VisiCalc offre, mentre tutti gli altri spiegano dettagliatamente *come* fargli svolgere i vari compiti. Prima di entrare nei particolari, è importante immaginare gli infiniti usi ai quali si può adibire il VisiCalc nella vita quotidiana.

Come tutti i buoni strumenti, su computer o di altro genere, il programma VisiCalc è tanto più prezioso quanto meglio viene utilizzato. Questo capitolo suggerisce alcuni impieghi elementari del VisiCalc e diverse applicazioni più avanzate. Per il momento conviene immaginare di conoscere il programma e di essere in grado di cominciare a considerarne le applicazioni; non ci si deve ancora preoccupare di *come* usarlo, ma solo concentrarsi su *cosa* fargli fare.

LE POSSIBILITA' DEL VISICALC

Si pensi a un qualsiasi insieme di numeri che si sia mai utilizzato, compilato o con il quale si abbia semplicemente avuto a che fare, sul lavoro o in casa. I numeri possono rappresentare qualsiasi cosa, da informazioni finanziarie riguardanti la famiglia, a registrazioni contabili, a dati scientifici o statistici: di solito questi numeri sono disposti in righe e colonne, come nella Figura 1.1. In assenza di scritte esplicative o di un titolo, i numeri sono privi di significato; tuttavia il loro formato a matrice costituisce già un'informazione: è chiaro che le righe e le colonne rappresentano categorie di qualche genere e che esiste un tema o un motivo per il quale i numeri sono disposti in una tabella.

Data una tabella come questa, tre sono i tipi di operazioni necessarie per rendere i numeri utili e significativi.

Primo, può darsi che vadano effettuati calcoli basati sui numeri della tabella. I risultati possono portare a un riepilogo dei valori tramite totali o medie; all'individuazione di trend statistici; alla costruzione di tabelle completamente diverse, basate sui numeri di quella iniziale; o addirittura alla risposta a domande o al raggiungimento di conclusioni in base ai dati.

11.18	6.35	8.13	3.56	2.54	0.76	1.52	2.03	1.78	3.81	7.62	9.40
7.62	7.11	5.59	3.30	0.25	0.00	0.00	0.00	0.51	0.76	5.08	5.59
11.18	7.62	6.35	4.06	1.02	0.25	0.00	0.00	0.51	2.54	5.84	10.16
1.52	1.78	3.05	4.83	6.86	4.83	4.57	3.30	2.79	2.79	2.03	1.02
4.83	5.33	7.62	9.91	9.91	11.18	9.40	7.37	7.37	7.11	6.35	5.08
11.43	11.68	13.97	10.67	10.67	11.94	17.02	13.46	14.22	5.84	9.91	12.95
6.60	5.59	7.87	8.89	8.89	8.38	3.89	7.87	7.11	6.60	7.11	6.10
5.59	5.08	5.33	9.30	15.49	22.86	17.53	17.24	22.10	20.83	6.86	4.06
6.60	6.35	8.38	7.37	9.55	8.89	10.41	12.09	7.87	6.86	7.37	7.62
6.86	7.37	9.40	8.38	8.89	7.62	9.40	10.16	8.38	7.37	9.65	8.89

Figura 1.1 - Gruppo di numeri disposti per righe e colonne

Secondo, è probabile che si vogliano presentare i dati in modo che il loro significato e la loro organizzazione risultino immediatamente evidenti a chiunque guardi la tabella. Questo lo si può ottenere tramite scritte che descrivano le righe e le colonne, dando un titolo alla tabella e forse includendo un'indicazione delle unità o della scala dei valori. Può darsi che sia necessario arrotondare i numeri, a seconda di quello che rappresentano, riducendo le cifre decimali. Infine, si può addirittura voler costruire una rappresentazione grafica dei valori, per esempio un istogramma, per rendere la raffigurazione più chiara e di maggior effetto. I destinatari di questa presentazione possono essere soci di affari, collaboratori in un progetto, insegnanti o compagni di classe, parenti o amici.

La terza operazione fondamentale è l'*immagazzinamento* dei numeri. Dev'essere possibile conservare queste informazioni numeriche in modo accurato e affidabile, così da poterle ritrovare rapidamente. Oppure, in una situazione più complicata, può essere necessario suddividere la tabella in più parti da immagazzinare separatamente, piuttosto che memorizzare la tabella tutta insieme. Può darsi che ciascuna parte abbia uno scopo distinto, così che sarebbe molto comodo poter suddividere i dati a seconda delle necessità.

Il programma VisiCalc è uno strumento che permette di effettuare queste tre operazioni con straordinaria semplicità e tuttavia con affidabilità assoluta. Attività che richiederebbero giorni se svolte a mano, o molte ore su una calcolatrice, con il VisiCalc vengono portate a termine in pochi minuti. Non è necessario essere programmatori per capire come funziona il VisiCalc; in effetti, lo si può cominciare a utilizzare praticamente non appena lo si è lanciato, ancor prima di capire tutto quello che offre. D'altro canto, quanto più lo si conosce, tanto più efficacemente lo si può usare in combinazioni con gli altri strumenti disponibili sul personal computer.

Quando si lancia il VisiCalc per la prima volta, si vede un grande reticolo di righe e colonne vuote, con centinaia di posizioni nelle quali è possibile inserire numeri, parole (chiamate *labels*, etichette) o formule matematiche. Il programma è stato costruito per offrire il massimo controllo sulle informazioni che vengono immesse

nelle righe e nelle colonne. Poichè è impossibile vedere contemporaneamente *tutte* le colonne e *tutte* le righe, lo schermo del computer diventa una "finestra" che si può spostare in alto o in basso sulle colonne o a destra o a sinistra sulle righe. Muovendo la finestra sulle informazioni registrate, si può guardare l'informazione desiderata in una particolare occasione.

Per fare qualche esempio di come il VisiCalc può essere utilizzato per effettuare le tre operazioni fondamentali a cui abbiamo accennato - calcolo, presentazione e immagazzinamento - esaminiamo cosa si può fare con i numeri della Figura 1.1. Una tabella di numeri ha un significato solo quando le vengono attribuite scritte esplicative (etichette) e un titolo, quindi creeremo tre situazioni immaginarie, ma plausibili, in cui si possono utilizzare quei numeri. In altri termini, attribuiremo tre significati diversi allo stesso insieme di numeri. Sarà forse necessario rinunciare in parte allo scetticismo, ma verrà illustrato un punto essenziale: il significato dei numeri non ha alcuna importanza per il VisiCalc, il quale si limita a effettuare le istruzioni e le operazioni che l'utente specifica.

Le tavole delle Figure 1.2, 1.3, 1.4 e 1.5 sono state costruite tutte con il VisiCalc; spesso tali tavole vengono chiamate *fogli elettronici* o *fogli di lavoro elettronici*, ma le si può chiamare come si vuole. Nell'esaminarle, si noterà la gamma dei cambiamenti apportati alle dieci righe e dodici colonne di numeri originali, e tuttavia la preparazione di ciascuno di questi fogli elettronici non ha richiesto che pochi minuti. Nei capitoli successivi verrà spiegato come costruire fogli elettronici di questo genere per far fronte alle diverse esigenze. Terminata la lettura di questo libro, si saprà esattamente come impartire istruzioni al programma VisiCalc perchè svolga tutte le operazioni presentate in questo capitolo, ma per ora non ci si preoccupi di come far funzionare il VisiCalc e si pensi solo alle sue possibili utilizzazioni.

IL VISICALC IN CASA

Cominciamo con una semplice applicazione domestica. Supponiamo di domandarci dove vanno a finire i soldi della spesa e di decidere di registrare gli acquisti per un periodo di dodici settimane. Per far questo, si sono divise le spese in dieci categorie (varie, cereali, latte e uova, cibo per il gatto e così via) e ogni settimana si registrano le somme spese per ogni categoria. (Sono state conservate le categorie dell'esempio originale americano e le somme sono espresse in dollari. *N.d.T.*) Siamo così arrivati alla fine del periodo di dodici settimane e si vuole un riepilogo delle spese. Specificatamente, si desidera conoscere:

- la spesa totale di ogni settimana e la spesa totale per ogni categoria per l'intero periodo di dodici settimane;

- la spesa percentuale per ogni categoria per l'intero periodo;

- la spesa media settimanale per ogni categoria.

Si lancia il VisiCalc e si inseriscono i numeri registrati per le dodici settimane, includendo un titolo per il foglio elettronico che si sta creando ed etichette per le categorie e le settimane. Successivamente, scrivendo alcune semplici formule nel formato che il VisiCalc comprende, si ottengono rapidamente tutte le informazioni desiderate. La Figura 1.2 mostra un loro possibile formato.

SPESE SETTIMANALI
DAL 30 MAGGIO AL 21 AGOSTO

	SETT. #1	SETT. #2	SETT. #3	SETT. #4	SETT. #5	SETT. #6	SETT. #7	SETT. #8
MISC.	11.18	6.35	8.13	3.56	2.54	0.76	1.52	2.03
CEREALI	7.62	7.11	5.59	3.30	0.25	0.00	0.00	0.00
LATTERIA	11.18	7.62	6.35	4.06	1.02	0.25	0.00	0.00
GATTO	1.52	1.78	3.05	4.83	6.86	4.83	4.57	3.30
VARIE	4.83	5.33	7.62	9.91	9.91	11.18	9.40	7.37
VEGET.	11.43	11.68	13.97	10.67	10.67	11.94	17.02	13.46
SIGAR.	6.60	5.59	7.87	8.89	8.89	8.38	3.89	7.87
CARNE	5.59	5.08	5.33	9.30	15.49	22.86	17.53	17.24
DOLCI	6.60	6.35	8.38	7.37	9.55	8.89	10.41	12.09
LIQUORI	6.86	7.37	9.40	8.38	8.89	7.62	9.40	10.16
**TOTALI	73.41	64.26	75.69	70.27	74.07	76.71	73.74	73.52

	SETT.#9	SETT.#10	SETT.#11	SETT.#12	TOTALI	%	MEDIE
MISC.	1.78	3.81	7.62	9.40	58.68	7	4.89
CEREALI	0.51	0.76	5.08	5.59	35.81	4	2.98
LATTERIA	0.51	2.54	5.84	10.16	49.53	6	4.13
GATTO	2.79	2.79	2.03	1.02	39.37	5	3.28
VARIE	7.37	7.11	6.35	5.08	91.46	11	7.62
VEGET.	14.22	5.84	9.91	12.95	143.76	17	11.98
SIGAR.	7.11	6.60	7.11	6.10	84.90	10	7.08
CARNE	22.10	20.83	6.86	4.06	152.27	18	12.69
DOLCI	7.87	6.86	7.37	7.62	99.36	12	8.28
LIQUORI	8.38	7.37	9.65	8.89	102.37	12	8.53
**TOTALI	72.64	64.51	67.82	70.87	857.51	100	71.46

Figura 1.2 - Applicazione domestica dei numeri

Naturalmente si deve poi decidere cosa fare di queste informazioni; il VisiCalc non può ridurre le spese. Il suo compito è di mostrare i fatti e le cifre rapidamente e in modo semplice, affinchè chi lo utilizza possa concentrarsi sulle decisioni da prendere.

IL VISICALC IN UFFICIO

L'applicazione successiva si riferisce a una situazione commerciale. Supponiamo di essere il direttore delle vendite di una piccola società con dieci venditori e di aver raccolto dati sul rendimento di ognuno di essi durante gli ultimi dodici mesi. Al termine dell'anno, si vuol vedere chi ha prodotto di più; inoltre si devono calcolare le retribuzioni e le commissioni totali, ed è arrivato il momento di dare premi di produzione.

Dopo aver lanciato il VisiCalc e aver cominciato a inserire i dati sulle vendite mensili di ogni addetto, ci si rende conto che si dovrà forse finire per licenziarne un paio dei meno produttivi. Questa decisione andrà giustificata al direttore generale e il modo più spettacolare di presentare il quadro complessivo è tramite un istogramma delle vendite annuali totali di ogni venditore.

La Figura 1.3 mostra l'inizio del foglio elettronico che si sta creando. I nomi dei venditori sono sulla sinistra, i mesi da gennaio a dicembre in alto, e il titolo spiega la funzione della tavola. Si noti come i valori delle vendite siano espressi in migliaia di dollari (questa convenzione semplifica l'immissione dei dati senza pregiudicarne troppo l'accuratezza.)

L'istruzione successiva impartita al VisiCalc è di calcolare le vendite mensili totali (addizionando i valori delle colonne) e le vendite totali per ogni venditore (addizionando i valori delle righe). Poi, battendo pochi tasti, si costruisce un istogramma basato su questi ultimi totali.

Dopo vanno calcolate le retribuzioni. I venditori hanno uno stipendio mensile di base di $1000 più una commissione del 5% sulle vendite. Utilizzando questa formula, si creerà una colonna del foglio di "stipendio più commissioni".

Infine, vanno stabiliti i premi di produzione. Ogni venditore ne riceve uno alla fine dell'anno, a seconda delle vendite annuali complessive. La tabella dei premi è la seguente:

vendite annuali	**premio**
$ 0 a $ 49.999	$ 250
$ 50.000 a $ 74.999	$ 1.000
$ 75.000 a $ 99.999	$ 1.500
$ 100.000 a $ 124.999	$ 2.250
$ 125.000 a $ 149.999	$ 3.000
$ 150.000 o più	$ 3.500

Affinchè il VisiCalc possa accedere a questa tabella nel modo più semplice possibile, si inserisce una "tavola di ricerca." (La funzione LOOKUP (ricerca) verrà studiata nel Capitolo 4.) Facendo riferimento a questa tavola, il VisiCalc può facilmente creare una colonna dei premi e poi una colonna delle retribuzioni totali dei venditori.

VENDITE MENSILI PER VENDITORE
(MIGLIAIA DI DOLLARI)

	GEN	FEB	MAR	APR	MAG	GIU	LUG	AGO
	===	===	===	===	===	===	===	===
BAKER	11.18	6.35	8.13	3.56	2.54	0.76	1.52	2.03
SMITH	7.62	7.11	5.59	3.30	0.25	0.00	0.00	0.00
FLINT	11.18	7.62	6.35	4.06	1.02	0.25	0.00	0.00
BROWN	1.52	1.78	3.05	4.83	6.86	4.83	4.57	3.30
VERN	4.83	5.33	7.62	9.91	9.91	11.18	9.40	7.37
MARLOW	11.43	11.68	13.97	10.67	10.67	11.94	17.02	13.46
HARPER	6.60	5.59	7.87	8.89	8.89	8.38	3.89	7.87
FLEMING	5.59	5.08	5.33	9.30	15.49	22.86	17.53	17.24
NASH	6.60	6.35	8.38	7.37	9.55	8.89	10.41	12.09
WHITE	6.86	7.37	9.40	8.38	8.89	7.62	9.40	10.16

	SET	OTT	NOV	DIC
	===	===	===	===
BAKER	1.78	3.81	7.62	9.40
SMITH	0.51	0.76	5.08	5.59
FLINT	0.51	2.54	5.84	10.16
BROWN	2.79	2.79	2.03	1.02
VERN	7.37	7.11	6.35	5.08
MARLOW	14.22	5.84	9.91	12.95
HARPER	7.11	6.60	7.11	6.10
FLEMING	22.10	20.83	6.86	4.06
NASH	7.87	6.86	7.37	7.62
WHITE	8.38	7.37	9.65	8.89

Figura 1.3 - Applicazione commerciale

Il foglio elettronico completo appare nella Figura 1.4. Esaminando le retribuzioni totali e l'istogramma, ci si rende immediatamente conto di chi ha un buon rendimento e di chi no.

IL VISICALC SCIENTIFICO

Il terzo esempio di foglio elettronico si riferisce al campo scientifico. Supponiamo di lavorare a una ricerca per la quale sia necessario conoscere i dati statistici sulle precipitazioni in diverse dozzine di grandi città degli Stati Uniti. Si è cominciato a raccogliere i dati per alcune città e si sono organizzate le informazioni su una tavola del VisiCalc chiamata "Precipitazioni mensili normali (in centimetri)".

VENDITE MENSILI PER VENDITORE
(MIGLIAIA DI DOLLARI)

	GEN	FEB	MAR	APR	MAG	GIU	LUG	AGO
BAKER	11.18	6.35	8.13	3.56	2.54	0.76	1.52	2.03
SMITH	7.62	7.11	5.59	3.30	0.25	0.00	0.00	0.00
FLINT	11.18	7.62	6.35	4.06	1.02	0.25	0.00	0.00
BROWN	1.52	1.78	3.05	4.83	6.86	4.83	4.57	3.30
VERN	4.83	5.33	7.62	9.91	9.91	11.18	9.40	7.37
MARLOW	11.43	11.68	13.97	10.67	10.67	11.94	17.02	13.46
HARPER	6.60	5.59	7.87	8.89	8.89	8.38	3.89	7.87
FLEMING	5.59	5.08	5.33	9.30	15.49	22.86	17.53	17.24
NASH	6.60	6.35	8.38	7.37	9.55	8.89	10.41	12.09
WHITE	6.86	7.37	9.40	8.38	8.89	7.62	9.40	10.16
**TOTALS	73.41	64.26	75.69	70.27	74.07	76.71	73.74	73.52

	SET	OTT	NOV	DIC	TOTALI	VEN+COM	PREMIO	TOT VEN
BAKER	1.78	3.81	7.62	9.40	58.68	14.93	1.00	15.93
SMITH	0.51	0.76	5.08	5.59	35.81	13.79	0.25	14.04
FLINT	0.51	2.54	5.84	10.16	49.53	14.48	0.25	14.73
BROWN	2.79	2.79	2.03	1.02	39.37	13.97	0.25	14.22
VERN	7.37	7.11	6.35	5.08	91.46	16.57	1.50	18.07
MARLOW	14.22	5.84	9.91	12.95	143.76	19.19	3.00	22.19
HARPER	7.11	6.60	7.11	6.10	84.90	16.25	1.50	17.75
FLEMING	22.10	20.83	6.86	4.06	152.27	19.61	3.50	23.11
NASH	7.87	6.86	7.37	7.62	99.36	16.97	1.50	18.47
WHITE	8.38	7.37	9.65	8.89	102.37	17.12	2.25	19.37
**TOTALS	72.64	64.51	67.82	70.87	857.51			

```
BAKER    ******
SMITH    ****
FLINT    *****
BROWN    ****
VERN     *********
MARLOW   **************
HARPER   ********
FLEMING  ***************
NASH     **********
WHITE    **********
```

TABELLA PREMI

0.00	0.25
50.00	1.00
75.00	1.50
100.00	2.25
125.00	3.00
150.00	3.50

Figura 1.4 - L'aplicazione commerciale completa

Questa tavola è quella della Figura 1.5. (Riconosciuti i numeri? La tavola contiene gli stessi valori di quelle degli altri due esempi; sembrano diversi perchè sono stati arrotondati all'intero. Questa operazione di arrotondamento è stata effettuata tramite un solo comando di formato globale, che verrà studiato nel Capitolo 2.)

PRECIPITAZIONI MENSILI NORMALI
(IN CENTIMETRI)

	GEN	FEB	MAR	APR	MAG	GIU	LUG	AGO
	===	===	===	===	===	===	===	===
HONOLULU	11	6	8	4	3	1	2	2
LOS ANGL	8	7	6	3	0	0	0	0
SAN FRAN	11	8	6	4	1	0	0	0
DENVER	2	2	3	5	7	5	5	3
ST.LOUIS	5	5	8	10	10	11	9	7
NEW ORLN	11	12	14	11	11	12	17	13
CLEVELND	7	6	8	9	9	8	4	8
MIAMI	6	5	5	9	15	23	18	17
WASH DC	7	6	8	7	10	9	10	12
NEW YORK	7	7	9	8	9	8	9	10

	SET	OTT	NOV	DIC	TOTALI
	===	===	===	===	======
HONOLULU	2	4	8	9	59
LOS ANGL	1	1	5	6	36
SAN FRAN	1	3	6	10	50
DENVER	3	3	2	1	39
ST.LOUIS	7	7	6	5	91
NEW ORLN	14	6	10	13	144
CLEVELND	7	7	7	6	85
MIAMI	22	21	7	4	152
WASH DC	8	7	7	8	99
NEW YORK	8	7	10	9	102

Figura 1.5 - Applicazione scientifica

Il nostro compito nella ricerca è di continuare a raccogliere i dati per le altre città previste, per poi effettuare diverse operazioni. Si devono calcolare alcuni valori statistici in base ai dati raccolti (medie, varianze, deviazioni standard ecc). Si deve poi ordinare la tavola secondo due criteri: prima alfabetico, a seconda del nome delle città, poi in ordine discendente rispetto alla precipitazione annua. In altri termini, i dati vanno presentati due volte, con ogni presentazione destinata a un uso diverso.

Supponiamo di aver acquisito una discreta conoscenza del computer, così da sapere che il VisiCalc non è che uno dei potenti strumenti di risoluzione di problemi che il computer offre. Anche un altro di questi strumenti, un linguaggio di programmazione chiamato BASIC, è facile da usare e può essere utilizzato insieme al VisiCalc per svolgere alcune operazioni in modo più rapido ed efficiente di quanto non sarebbe possibile con uno solo dei due strumenti.

Dopo aver riflettuto, si decide che il modo migliore per effettuare queste opera-

zioni - il calcolo dei valori statistici, l'ordinamento dei dati e la presentazione delle tavole delle precipitazioni in forma pulita e ordinata - consiste nell'utilizzare le capacità combinate del VisiCalc e del BASIC. Il VisiCalc è l'ideale per operazioni di *input* e di *output* - cioè per inserire i dati nel computer e per preparare la presentazione finale dei dati. Tuttavia alcuni calcoli statistici, e senz'altro le operazioni di ordinamento, possono rivelarsi un pò ingombranti per il VisiCalc, mentre si sa che esistono programmi molto facili in BASIC, che svolgono questi calcoli in modo semplice ed elegante.

L'unico problema è: come si fa a trasferire i dati sulle precipitazioni dal VisiCalc al BASIC? E' possibile rendere i dati disponibili a più di uno strumento utilizzabile sul computer?

Ancora una volta il VisiCalc offre una soluzione a un problema, fornendo un metodo per immagazzinare i dati di un foglio elettronico in un file su disco di tipo particolare, chiamato file DIF. DIF sta per *Data Interchange Format* (formato per l'interscambio dei dati). Il motivo per cui il DIF è così speciale è che immagazzina i dati in modo tale da renderli leggibili al programma VisiCalc o a qualunque programma in BASIC. In breve, i file DIF forniscono un metodo per lo scambio di dati fra i diversi strumenti di programmazione che si può aver modo di utilizzare. L'importanza e l'uso dei file DIF verranno esaminati negli ultimi tre capitoli del libro.

In breve, ecco un riepilogo dei passaggi necessari per svolgere le operazioni per la costruzione della tavola delle precipitazioni:

- Innanzi tutto si userà il VisiCalc per costruire una tavola come quella della Figura 1.5. Quando si è pronti per le operazioni intermedie - il calcolo delle statistiche e l'ordinamento dei dati - si dirà al VisiCalc di creare un file DIF contenente i dati sulle precipitazioni.

- Successivamente si utilizzeranno programmi in BASIC in grado di leggere il file DIF e di effettuare le operazioni di calcolo e di ordinamento. I Capitoli 7 e 8 contengono due programmi completi in BASIC per svolgere questi compiti e tutte le informazioni necessarie per l'uso dei programmi. (Inoltre l'Appendice B contiene una breve introduzione al BASIC, per chi vi si avvicina per la prima volta.)

- Si userà poi uno di questi due programmi in BASIC (quelli del Capitolo 8) per creare un nuovo file DIF con i dati sulle precipitazioni *ordinati*. Il passaggio finale sarà di istruire il VisiCalc per l'inserimento del nuovo file DIF nel foglio elettronico, così da ottenere esattamente il formato dei dati desiderato.

Il punto essenziale è il seguente: il programma VisiCalc consente di utilizzare i dati del foglio elettronico con altri strumenti di programmazione disponibili sul computer, il che permette di sfruttare le possibilità di questi programmi per la soluzione di compiti complicati.

UN'OCCHIATA AL FUTURO

Abbiamo esaminato tre diverse situazioni in cui il VisiCalc si rivela un utile strumento per il calcolo, la presentazione e l'immagazzinamento di grandi tabelle di numeri. La chiave per ottenere tutte queste cose è un insieme di semplici comandi battuti sulla tastiera. Questi comandi comunicano al VisiCalc esattamente quali sono le operazioni da effettuare sui numeri inseriti nel foglio elettronico. Come tutti gli altri aspetti del VisiCalc, il modo in cui vengono impartiti i comandi è innanzi tutto caratterizzato dalla semplicità: il VisiCalc fornisce un *rigo dei comandi* che mostra le opzioni disponibili per la manipolazione del foglio elettronico. Quando si usa il VisiCalc, si può esaminare questo rigo dei comandi battendo sulla tastiera il tasto della barra (/). Nella parte superiore dello schermo comparirà:

COMMAND: BCDEFGIMPRSTVW-

Ogni carattere del rigo dei comandi (le lettere e il carattere "-") corrisponde a una funzione del VisiCalc. Per utilizzarle, basta premere sulla tastiera il tasto adeguato mentre sullo schermo è visibile il rigo dei comandi.

Quindi, per imparare a usare il programma VisiCalc, in pratica basta apprendere l'azione di ciascun comando, che è quanto verrà spiegato nel prosieguo del libro. La Figura 1.6 contiene un breve riepilogo delle funzioni dei comandi; può convenire darle un'occhiata adesso, così da farsi un'idea generale sulle possibilità offerte dal VisiCalc. Non si deve però cercare di memorizzare le istruzioni a questo punto; in seguito, quando si sarà acquisita una certa esperienza nell'uso del programma, la Figura 1.6 potrà essere un'utile tavola di consultazione.

Per fare un riepilogo dei comandi del VisiCalc, li si può dividere in tre gruppi principali a seconda della funzione che svolgono - calcolo, controllo e memorizzazione. La principale funzione di calcolo è costituita dal comando di replica, /R. I comandi di controllo permettono di *cancellare* le informazioni dal foglio elettronico (/B, /C, /D), di *modificare* o di *riformattare* le informazioni (/E, /F, /G, /I, /M) e di *controllare* la porzione del foglio elettronico che appare sullo schermo (/W, /T). Infine, i comandi di memorizzazione comprendono sia la memorizzazione su disco (/S) che l'output a una stampante (/P).

/B B sta per *blank* (vuoto). Il comando consente di cancellare il contenuto di qualsiasi posizione del foglio, che si tratti di un'etichetta o di un valore numerico.

/C Il comando *cancella* tutto lo schermo e fornisce un foglio vergine.

Figura 1.6 - Breve riepilogo dei comandi del VisiCalc (continua)

/D Questo comando consente di cancellare (*delete*) tutta una riga o tutta una colonna del foglio.

/E Questo comando permette di modificare (*edit*) il contenuto di qualsiasi posizione dello schermo. (Alcune versioni del VisiCalc non hanno questa possibilità.)

/F Il comando di *formattazione* permette di indicare come si vuole che i dati siano rappresentati sul foglio del VisiCalc. Abbiamo visto che i dati numerici possono essere rappresentati in forma decimale o intera. Le informazioni possono anche essere giustificate a destra o a sinistra nell'ambito di una posizione. Il comando di formattazione serve anche per creare istogrammi.

/G G sta per *globale*. Questo comando consente di formattare tutto lo schermo del VisiCalc, piuttosto che una sola posizione del foglio. Il comando globale può essere utilizzato anche per aumentare il numero degli spazi di ogni colonna del foglio. (/G offre anche alcune possibilità che riguardano i calcoli numerici.)

/I Questo comando consente di *inserire* una riga o una colonna in qualsiasi punto del foglio.

/M Con questo comando si può spostare (*move*) qualsiasi riga o colonna in un'altra posizione del foglio.

/P Questo comando permette di stampare (*print*) su carta il foglio.

/R Questa è forse la possibilità più importante che il VisiCalc offre e probabilmente è anche il comando che verrà utilizzato più spesso. R sta per *replica*: quando si è scritto una formula aritmetica nel VisiCalc, /R consente di applicarla istantaneamente a intere colonne e righe di dati. Questo comando è l'essenza del programma VisiCalc.

/S Il comando di memorizzazione (*storage*) consente di memorizzare su disco sia file del VisiCalc che file DIF e di riportarli dal disco al VisiCalc.

Figura 1.6 - Breve riepilogo dei comandi del VisiCalc (continua)

/T Con questo comando si possono bloccare righe o colonne, che quando si muoverà la finestra non si sposteranno. Normalmente queste righe e queste colonne contengono etichette, o *titoli*, che si vuol avere sempre in vista quando ci si sposta sul foglio.

/V Comunica con quale *versione* del VisiCalc si sta lavorando.

/W Consente di dividere lo schermo in due finestre (*windows*) per poter vedere contemporaneamente due parti del foglio.

/— Questo comando consente di tracciare linee sul foglio per una presentazione più chiara delle informazioni. Ognuna delle tavole delle Figure da 1.2 a 1.5 ha una linea doppia che separa il titolo dal resto dei dati. Queste linee sono state tracciate con il comando /—.

Figura 1.6 - Breve riepilogo dei comandi del VisiCalc (fine)

VERSIONI DEL VISICALC

Il programma VisiCalc è stato progettato per essere utilizzato su molti microcomputer e alcuni particolari variano a seconda della macchina di destinazione. Spesso le differenze dipendono più dal computer che dal VisiCalc: per esempio, l'aspetto della "finestra" sul foglio varia. Il numero di colonne e di righe che possono apparire simultaneamente sullo schermo e la forma del *cursore* non sono uguali su tutti i computer. Il cursore, come vedremo nel Capitolo 2, è un indicatore di posizione che contrassegna il punto del foglio in cui vengono effettuate le operazioni. Lo si può spostare in quattro direzioni - in alto, in basso, a sinistra, a destra - e il modo per far questo cambia a seconda del computer; alcuni hanno quattro tasti direzionali, altri solo due, per i quali si deve scegliere fra modalità orizzontale e verticale.

Comunque questo genere di variazioni non rende le versioni del VisiCalc e *essenzialmente* diverse le une dalle altre, solo che quando si inizia a utilizzare il VisiCalc su un computer si deve sapere con cosa si ha a che fare.

Alcune variazioni, invece, si riferiscono a possibilità che certe versioni del VisiCalc non offrono. Per esempio, talvolta il comando di editing (/E) non è disponibile certe versioni non sono dotate di *funzioni* che verranno trattate nel Capitolo 4. Quando esamineremo queste caratteristiche cercheremo di analizzare metodi alternativi per svolgere quelle operazioni.

La differenza più rilevante è che alcune versioni del VisiCalc non consentono la creazione di file DIF, nel qual caso non si può generare un file di tale tipo per utilizzare i dati con altri programmi. Per fortuna questa deficienza è rara.

MEMORIA

Di solito non è necessario preoccuparsi del modo in cui il computer memorizza un foglio elettronico nella sua memoria interna. Lo si deve fare solo quando lo spazio comincia a scarseggiare. Lo schermo del VisiCalc dà un'indicazione della memoria disponibile durante la creazione di un foglio l'indicatore è il numero intero nell'angolo in alto a destra dello schermo e dà il numero di *kilobyte* di memoria che rimangono. (Un *byte* è lo spazio di memoria necessario per la memorizzazione di un carattere; un kilobyte è 1024 byte.) Quando si lancia il VisiCalc, il numero può essere grande o piccolo a seconda delle dimensioni della memoria del computer; via via che si costruisce il foglio elettronico, il numero diventa sempre più piccolo, dato che si sta occupando parte della memoria disponibile. Se il computer ha una memoria piccola, si potranno costruire solo fogli elettronici piccoli, con relativamente pochi dati. Nel Capitolo 6 verrà presentata una tecnica per superare in parte i limiti di una piccola memoria centrale. Questo metodo implica la memorizzazione di sezioni di fogli elettronici in file DIF, che verranno poi riuniti in nuovi fogli elettronici.

Quando si è utilizzata tutta la memoria, il numero dell'indicatore di memoria è sostituito dalle lettere "OM".

IL VISICALC E IL BASIC

Finora abbiamo visto come il VisiCalc liberi dal tedio di molte operazioni ripetitive di elaborazione dei dati e consenta di creare tabelle e grafici molto più rapidamente che non a mano. Ma il grande fascino dell'utilizzazione del VisiCalc è la sua semplicità. Forse uno dei motivi principali dell'attrattiva che il programma esercita su molti utenti di personal computer è il fatto che non è necessario conoscere un linguaggio di programmazione.

Questo significa che uno strumento come il VisiCalc può sostituire completamente i linguaggi di programmazione dei microcomputer, come il BASIC o il Pascal? Naturalmente no; però una domanda rimane legittima: data la varietà di strumenti di programmazione disponibili sul computer, come si fa a decidere se utilizzarne uno piuttosto che un altro?

Un aspetto di cui bisogna tener conto è il tipo di computer che si sta utilizzando: un *personal* computer. Questo significa che la sua utilizzazione per risolvere problemi non è definita nè dai costruttori, nè dagli autori del software, ma dall'utente. Il computer è controllato dall'intelligenza di chi lo usa, dalla sua creatività e ingegnosità e dalle sue preferenze sul modo migliore di gestire una particolare operazione. (Inoltre, di solito il computer reagisce con notevole avidità agli errori dell'utente.)

Poichè chi usa il computer lo controlla, è chiaramente un vantaggio avere a propria disposizione quanti più strumenti possibili. E' vero che si possono fare molte cose avvincenti con il solo VisiCalc, ma se ne possono fare di più con il VisiCalc e il BASIC. Questo sarà uno dei temi ricorrenti del libro: l'apprendimento

del VisiCal è divertente e remunerativo, ma è ancor più importante imparare a inserire il VisiCalc nel contesto dell'utilizzazione del personal computer.

Chi non sa nulla del BASIC non si preoccupi: si conquisti una provincia per volta e, senza accorgersene, si sarà conquistato tutto il regno.

Si può leggere, capire e *utilizzare* praticamente tutto il contenuto di questo libro senza nessuna conoscenza del BASIC. Negli ultimi tre capitoli presenteremo alcuni programmi in BASIC che leggono e creano file DIF e quindi comunicano indirettamente con il VisiCalc. Questi programmi possono essere utilizzati senza che se ne capisca completamente il funzionamento, oppure li si può studiare e modificare, adattandoli alle necessità. Nel frattempo l'Appendice B fornisce un breve compendio del BASIC con qualche annotazione sulle differenze fra l'effettuare un'operazione con il BASIC e l'effettuare la stessa operazione tramite il VisiCalc. Nel leggere questa descrizione della programmazione in BASIC, viene naturale mettere il BASIC a confronto con il programma VisiCalc. Quest'ultimo svolge l'utile compito di rendere molti aspetti della programmazione completamente invisibili all'utente: quando si utilizza il VisiCalc non si deve pensare alla relazione fra l'immagazzinamento dei dati e la logica del programma e per ripetere calcoli basta battere "/R". Per fare un esempio forse più pertinente, *nel VisiCalc le operazioni di input e di output sono veramente simultanee*, cioè non si deve inserire un gruppo di dati e poi pensare a come rappresentarli: la tavola di output viene creata nel momento in cui i dati vengono immessi.

D'altro canto, quanto più si utilizza il VisiCalc, tanto più è probabile che si incontrino situazioni in cui il programma non risponde esattamente alle necessità, o nelle quali l'uso del VisiCalc sembra rendere il compito più complesso, non più semplice. Tale fatto non diminuisce assolutamente il valore del VisiCalc: un falegname non butta via la sega perchè si accorge che non la può usare per piantare i chiodi. Tutto diventa più facile una volta imparato a scegliere bene gli strumenti. Avanzando nello studio di questo libro e acquistando sempre più dimestichezza con la meccanica del VisiCalc, una cosa va sempre tenuta presente: quali sono le applicazioni più consone alle caratteristiche del VisiCalc e quali si addicono più a un altro strumento disponibile sul computer o a più strumenti combinati.

IL SISTEMA OPERATIVO

Le funzioni essenziali del computer sono controllate da una struttura centrale di software, chiamata *sistema operativo*. Tale sistema gestisce l'input e l'output ha il controllo dei file su disco carica i programmi nella memoria centrale del computer e li lancia infine controlla il software che traduce i linguaggi di programmazione (come il BASIC e il Pascal) nel codice macchina che il computer è in grado di eseguire.

Quando si inserisce il dischetto con il VisiCalc nel drive e si accende il computer, possono succedere due cose, a seconda della versione del programma: o il sistema operativo carica il VisiCalc e lo lancia, così che compare subito la finestra del

VisiCalc, oppure il sistema operativo genera un messaggio indicante che sta aspettando un comando. In quest'ultimo caso, si deve impartire il comando che comunica al sistema operativo di caricare il VisiCalc e di lanciarlo. Spesso questo comando sarà semplicemente il nome con cui il programma VisiCalc è memorizzato.

Il comando del VisiCalc /S offre un'opzione per uscire dal programma e tornare al livello dei comandi del sistema operativo. Ci possono essere molti motivi per uscire in tal modo dal VisiCalc: per effettuare operazioni su disco che non possono essere svolte dall'interno del VisiCalc; per usare un editor; per scrivere o lanciare un programma in BASIC o in qualche altro linguaggio di programmazione. E' importante ricordarsi che quando si esce dal VisiCalc si perde il foglio elettronico al quale si lavorava. Per questa ragione è bene memorizzare il foglio elettronico su disco prima di abbandonare il programma. Anche questo può essere effettuato con il comando /S, che verrà dettagliatamente esaminato nel Capitolo 2.

ALTRI PROGRAMMI DI FOGLIO ELETTRONICO

Dopo l'avvento del VisiCalc, sul mercato sono comparsi molti altri programmi di foglio elettronico concorrenziali. Alcuni sono progettati per computer specifici o per operare su certi sistemi operativi. Per esempio, diversi operano su CP/M, uno dei sistemi più diffusi su personal computer. Come in altri campi del software per microcomputer, la grande varietà dei fogli elettronici in commercio può confondere l'acquirente potenziale; la possibilità di scelta è molto ampia e prendere una decisione oculata può essere difficile.

Ci sono comunque diverse domande di carattere generale da porre prima di comprare un programma di foglio elettronico per un personal computer. Quali sono le difficoltà di apprendimento? Sembra che fornisca tutte le funzioni necessarie per l'utilizzazione alla quale dev'essere adibito? Ha possibilià grafiche? Comunica con altri programmi? In particolare, crea file che possano essere facilmente letti da un programma in BASIC? Se il computer ha un programma di word processing, è facile incorporare i fogli elettronici in un testo dello word processor? A chi non ha ancora comprato un programma di foglio elettronico, durante la lettura di questo libro verranno senz'altro in mente altre domande.

SOMMARIO

Il programma VisiCalc è uno strumento che consente di eliminare per sempre dalla propria vita il tedio delle operazioni aritmetiche su grandi insiemi di numeri. Il programma è basato su righe e colonne nelle quali possono essere inseriti numeri, etichette o formule. Nel VisiCalc l'input e l'output costituiscono un'unica operazione; una tavola su foglio elettronico viene creata disponendo i dati esattamente dove li si vogliono. Se si indicano calcoli, il VisiCalc li effettua istantaneamente.

Il VisiCalc viene controllato tramite una dozzina di comandi monoletterali che compaiono quando si batte il tasto "/": è questa la possente semplicità del programma, che ha portato alla sua enorme popolarità.

Il programma VisiCalc è stato intelligentemente progettato in modo da operare in connessione con altri strumenti di software, quindi la capacità di creare file DIF costituisce una delle sue caratteristiche essenziali.

Nel Capitolo 2 ci tufferemo nei particolari del funzionamento del VisiCalc. Per chi già lo possiede, è arrivato il momento di lanciarlo e di seguire il libro.

CAPITOLO 2

UN'INTRODUZIONE AL VISICALC

LA FINESTRA SUL FOGLIO ELETTRONICO

Quando si lancia il VisiCalc e si preme il tasto "/" per ottenere il rigo dei comandi, sullo schermo compare qualcosa di simile al contenuto della Figura 2.1. Innanzi tutto si nota una colonna di numeri, che partono da 1, sulla sinistra dello schermo, e una riga di lettere, che partono da A e sono distanziate in modo regolare, nella parte superiore dello schermo. I numeri identificano le righe, le lettere le colonne. (Il numero di righe e di colonne che compaiono sullo schermo dipende dal computer; possono essere più o meno di quelle della Figura 2.1.)

Associando una lettera a un numero si può indicare qualsiasi posizione del

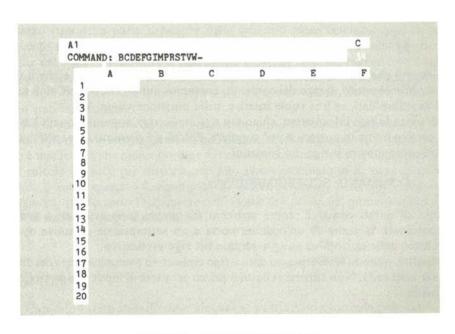



Figura 2.1 - Lo schermo del VisiCalc

foglio; queste coordinate possono essere considerate l'"indirizzo" della posizione. Come le posizioni dell'ufficio e della casa possono essere definite come l'intersezione fra due strade (corso Italia e via Garibaldi, via Cavour e via Lamarmora), così una posizione del foglio del VisiCalc può essere identificata dall'intersezione fra una colonna e una riga. Nell'indirizzo la lettera della colonna precede sempre il numero; per esempio la posizione all'intersezione fra la colonna C e la riga 9 è chiamata C9.

Quando il VisiCalc viene lanciato, la posizione A1, quella in alto a sinistra del foglio elettronico, contiene un segno grafico che la distingue dalle altre visibili sullo schermo. A seconda delle versioni del VisiCalc, tutta la posizione A1 è messa in risalto da un rettangolo chiaro, oppure è racchiusa da due grandi parentesi quadre. Quale che sia la rappresentazione grafica, si tratta del *cursore* del VisiCalc. Il cursore indica la "posizione corrente"; in altri termini, se si inseriscono dati nel foglio elettronico, questi vengono immagazzinati nella posizione indicata dal cursore.

Al di sopra del foglio elettronico ci sono tre righi informativi che non scompaiono mai (anche se talvolta sono vuoti), qualunque sia la posizione del cursore. I tre righi forniscono informazioni essenziali sul foglio elettronico, quindi saperli leggere correttamente è importante.

Il primo rigo, che chiameremo *rigo dei contenuti*, mostra l'indirizzo e il contenuto della posizione corrente. All'inizio di una sessione di lavoro con il VisiCalc, prima dell'inserimento dei dati, questo rigo indica semplicemente l'indirizzo corrente del cursore:

A1

Si ricordi che quando si lancia il programma il cursore è in A1 e che il foglio elettronico è vuoto. Il rigo dei contenuti comunica quindi che il VisiCalc è pronto ad accettare dati, se li si vuole inserire, nella posizione vuota A1.

Il secondo rigo informativo, chiamato *rigo esplicativo*, comunica qual'è l'attività in corso o fra quali opzioni si può scegliere. Poichè si è premuto il tasto "/", il rigo esplicativo mostra il rigo dei comandi

COMMAND: BCDEFGIMPRSTVW-

Molti di questi comandi, come vedremo fra breve, sono associati a comandi subordinati: la scelta di un'opzione porta a un sottoinsieme di nuove opzioni. L'elenco delle opzioni va sempre cercato nel rigo esplicativo.

Inoltre, quando si inseriscono dati, il rigo esplicativo comunica il *tipo* dei dati che si sta inserendo. Non appena si batte il primo carattere di input, il rigo esplicativo mostra:

VALUE
(valore)

se si inseriscono dati numerici, o

LABEL
(etichetta)

se i dati inseriti non sono numerici. L'utilità di questa informazione può sembrare dubbia; dopo tutto si dovrebbe sapere se si sta inserendo un numero o una parola. Questo servizio ha però due scopi: prima di tutto aiuta a riconoscere la fonte di errori di input. Per esempio, si comincia a scrivere un numero e inavvertitamente vi si inserisce un carattere non numerico; il VisiCalc deve avere un metodo per avvertire della discrepanza. Alcuni sistemi, in questa situazione, fanno suonare un campanello, così che l'utente guarda automaticamente lo schermo per vedere in cosa ha sbagliato. La parola VALUE sul rigo esplicativo ricorda che è in corso l'inserimento di un numero e suggerisce che è stata innavvertitamente inserita una lettera o qualche altro carattere non numerico.

In alcune situazioni, il messaggio LABEL o VALUE ha anche un altro scopo. In certi casi si inserirà una cifra che dev'essere considerata un'etichetta e non un valore, mentre in altri si inserirà una lettera che rappresenta un valore. Il VisiCalc ha convenzioni per far fronte a entrambe le situazioni; talvolta, quando le si utilizzano, si desidera ricevere conferma di averle impiegate correttamente. Il VisiCalc la fornisce comunicando se sta accettando l'inserimento come valore o come etichetta. Nel prosieguo del capitolo questo punto sarà trattato dettagliatamente.

Il terzo rigo nella parte superiore dello schermo viene chiamato rigo di *edit* (intervento sui dati). Quando si inserisce un'etichetta o un valore nella posizione corrente del foglio elettronico, il rigo di edit mostra i caratteri battuti fino a quel momento. Se si impartisce il comando di edit, /E, le modifiche vengono effettuate su tale rigo. (Si noti come questo rigo sia al momento vuoto, dato che non sono ancora stati inseriti dati.)

Prima di spiegare come inserire i dati nel foglio elettronico, vanno notati due elementi informativi sempre presenti nell'angolo in alto a destra dello schermo. Il primo è una lettera che indica l'ordine di calcolo; inizialmente è una C, per indicare che il calcolo procede per colonne, ma può essere cambiata in R, per righe. Nel Capitolo 5 verrà spiegato il significato di questa funzione. Il secondo elemento informativo, subito sotto l'indicatore dell'ordine di calcolo, è il numero intero che indica la memoria ancora disponibile.

LA TASTIERA

Nel Capitolo 1 sono state discusse alcune delle variazioni a seconda delle versioni del VisiCalc sui diversi computer. La tastiera, che costituisce l'unico mezzo di comunicazione con il programma, può richiedere combinazioni particolari di tasti

per far svolgere alcune comuni operazioni. L'uso della tastiera diventa rapidamente automatico, una volta apprese le funzioni dei tasti.

In questo libro ad alcuni tasti vengono attribuiti nomi che possono non corrispondere a quelli di un particolare computer. Le *azioni* sono le stesse, ma i tasti che ne permettono l'effettuazione possono essere diversi. Ecco quindi un elenco dei nomi che daremo alle più comuni funzioni della tastiera:

- *Tasti direzionali: freccia verso l'alto, freccia verso il basso, freccia a sinistra, freccia a destra.* Sono i tasti che permettono di spostare il cursore: un computer può averne quattro o solo due. Nel secondo caso ci sarà un metodo per passare dal controllo dello spostamento orizzontale a quello dello spostamento verticale: e lo schermo del VisiCalc indicherà la direzione vigente.

- *Tasto escape.* Quando si inseriscono dati o si utilizza il comando edit (/E), questo tasto consente di cancellare caratteri dal rigo di edit. Ogni volta che si preme il tasto escape si cancella un carattere.

- *Tasto break.* E' un tasto che torna utile in molte situazioni: se si sta inserendo dati, ci si accorge di aver commesso un errore e si vuol cancellare l'intero contenuto del rigo di edit, il tasto break consente di farlo. Oppure, se si sta eseguendo uno dei comandi del rigo dei comandi e si cambia improvvisamente idea, il tasto break permette di uscire completamente dalla situazione, anche se si è invischiati in uno degli insiemi subordinati di opzioni. Un computer può avere un tasto break specifico, oppure può essere necessario premere contemporaneamente due tasti.

- *Tasto return.* E' quello utilizzato per inserire nel foglio elettronico i dati del rigo di edit. Nella tastiera di alcuni computer si chiama ENTER.

Prima di procedere nella lettura è bene individuare i tasti che permettono di effettuare queste funzioni. L'Appendice C dà una descrizione per i computer più comuni.

SPOSTAMENTO DEL CURSORE... CONTROLLO DELLA FINESTRA

Se lo schermo del VisiCalc è sempre simile a quello della Figura 2.1, premere il tasto break per eliminare il rigo dei comandi. Il cursore è ancora nell'angolo superiore sinistro del foglio elettronico e il rigo dei contenuti mostra l'indirizzo A1. Il rigo esplicativo e quello di edit sono vuoti.

Esaminiamo gli spostamenti del cursore. Premere la freccia a destra una volta. Sullo schermo succedono due cose: il cursore si sposta a destra di una posizione e il rigo dei contenuti mostra l'indirizzo della nuova posizione corrente, B1. Premere

adesso la freccia verso il basso. Il cursore si sposta alla posizione B2. Qualsiasi movimento del cursore ha due ripercussioni sullo schermo: la posizione fisica del cursore cambia e sul rigo dei contenuti compare il nuovo indirizzo.

Si sposti ora il cursore al bordo destro dello schermo, poi lo si porti all'angolo inferiore destro, premendo diverse volte la freccia a destra e quella verso il basso. Lo schermo sarà grosso modo come quello della Figura 2.2 (naturalmente l'indirizzo della posizione in basso a destra dipenderà dal numero di colonne e di righe che il VisiCalc utilizzato mostra.)

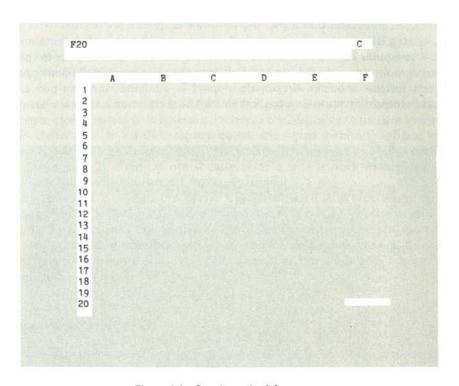



Figura 2.2 - Spostamento del cursore

Se si preme un'altra volta la freccia a destra, si vedrà qualcosa di nuovo: la *finestra* si sposta. Ci si accorge che si è spostata perchè le etichette delle colonne nella parte superiore del foglio elettronico si sono portate a destra di una posizione. Lo stesso succede per le righe se si preme la freccia verso il basso: la riga 1 scompare e ne compare una nuova in fondo al foglio. Questo illustra il concetto di finestra. Il foglio elettronico è grande, ma la porzione che se ne può esaminare è limitata dalle dimensioni dello schermo del computer. Il VisiCalc consente di "spostare" la finestra-schermo su qualsiasi parte del foglio elettronico, spostando semplicemente

il cursore; i dati che non appaiono sullo schermo non sono perduti, ma semplicemente invisibili.

(Se si preferisce, quanto avviene sullo schermo può essere considerato in un altro modo: quando la finestra si sposta sulla destra, il foglio elettronico scorre sulla sinistra, come se fosse avvolto su cilindri girevoli da entrambi i lati. Due sono le metafore con cui viene descritto quanto succede: o la finestra si sposta o il foglio scorre; significano tutte e due la stessa cosa).

Si continui a spostare a destra il cursore. Le colonne verranno indicate dalle lettere di tutto l'alfabeto, fino a Z, poi cominceranno a essere contrassegnate da *due* lettere, da AA ad AZ. Ma nemmeno questa è l'ultima colonna: la successiva è BA, e le colonne continuano fino a BK, per un totale di 63.

Arrivati a BK, si cominci a spostare il cursore verso il basso. (Incidentalmente, forse il computer ha una funzione di *ripetizione*, che rende più semplice lo spostamento di molte posizioni del cursore. Può darsi che ci sia un tasto di ripetizione, da premere insieme a quello direzionale, oppure la modalità ripetitiva può essere attivata tenendo premuto il tasto direzionale.) Se si continua a spostare il cursore

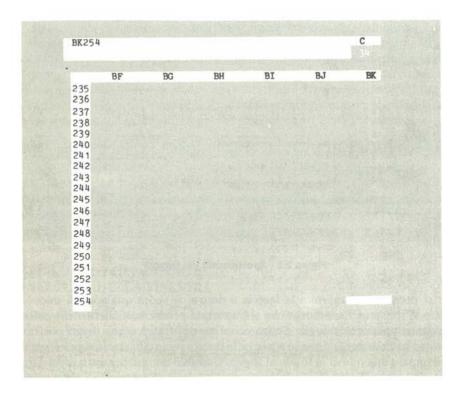



Figura 2.3 - Limiti del foglio elettronico

verso il basso, si finirà per raggiungere il limite del foglio elettronico del VisiCalc: l'indirizzo di questa posizione è BK254, come mostra la Figura 2.3.

Con una rapida moltiplicazione del numero delle righe per quello delle colonne, si vede che il foglio contiene un po' più di 16.000 posizioni. Si ricordi però che lo spazio utilizzabile del foglio dipende soprattutto dalla disponibilità di memoria del computer.

Torniamo adesso alla posizione A1. Fortunatamente il VisiCalc permette di effettuare istantaneamente lunghi salti sul foglio, senza doversi spostare posizione per posizione. Lo si può fare tramite il comando GO TO (vai a), che viene impartito battendo il simbolo "maggiore di" (>). Dopo che è stato battuto >, sul rigo esplicativo compare il messaggio:

GO TO:COORDINATE
(vai a:coordinata)

Coordinate è il termine con cui il VisiCalc designa quello che abbiamo chiamato indirizzo, cioè la lettera (o le lettere) e il numero che identificano una posizione sul foglio elettronico. Il VisiCalc aspetta quindi che venga fornito l'indirizzo della posizione alla quale dev'essere spostato il cursore. Si batta A1; l'indirizzo comparirà sul rigo di edit. Premere ora il tasto return: ci si troverà istantaneamente nell'angolo superiore sinistro del foglio elettronico.

Può darsi che, prima di procedere, si vogliano fare altri esperimenti con il comando GO TO. Per riassumere, abbiamo scoperto che il foglio elettronico del VisiCalc ha 63 colonne (da A a BK) e 254 righe; il cursore può essere portato a qualsiasi posizione o tramite i tasti direzionali o con il comando GO TO. Uno spostamento del cursore al di fuori della finestra corrente modifica automaticamente anche la finestra.

INSERIMENTO DEI DATI

Il VisiCalc consente tre diversi tipi di inserimento di dati, che però danno luogo a due tipi di dati soltanto. Le *etichette* rappresentano essenzialmente dati non numerici; i *valori* e i *riferimenti a valori* rappresentano dati numerici.

Inoltre, i riferimenti a valori possono essere utilizzati per creare *formule*. Più semplicemente, un riferimento a valore rappresenta un modo per duplicare il contenuto di qualsiasi posizione del foglio e utilizzarlo in un'altra.

Cominciamo con i tre tipi di inserimento.

Etichette

Se il cursore non è nella posizione A1, ve lo si porti con il comando GO TO. Come esempio di inserimento di etichetta, cominciamo a battere la parola VISI-

CALC e notiamo cosa succede a ogni stadio dell'operazione. Non appena si batte la prima lettera, V, sullo schermo si verificano quattro cambiamenti:

1. sul rigo esplicativo compare la parola LABEL, a indicare che è in corso l'inserimento di un dato non numerico;

2. la prima lettera dell'inserimento, V, compare nella posizione corrente, A1, che è contraddistinta dal cursore;

3. la V compare anche sul rigo di edit;

4. sul rigo di edit, dopo la V, ci sarà un segno grafico, una *cue* (indicatore, leggi "chiu"), a indicare che l'inserimento non è stato completato. (Ancora una volta, la forma della *cue* dipende dalla versione del VisiCalc: può essere un quadratino lampeggiante o semplicemente una lineetta.) Finchè la *cue* rimane visibile, si può cancellare l'inserimento premendo il tasto break o cancellare l'ultimo carattere tramite il tasto escape.

Così, per fotografare un momento dell'operazione, ecco il contenuto dei tre righi superiori dello schermo dopo aver battuto la V di VISICALC:

```
A1
LABEL
V□
```

Continuiamo a scrivere la parola. Tutte le volte che viene battuta una lettera, questa compare sia nella posizione del cursore che sul rigo di edit, e la *cue* sul rigo di edit si sposta a destra di uno spazio. Dopo aver battuto tutta la parola, premiamo il tasto return e guardiamo cosa succede. Tanto il rigo esplicativo che quello di edit tornano vuoti, comunicando che è stata completata l'operazione di inserimento. La parola VISICALC rimane nella posizione A1; finchè anche il cursore è in A1, il rigo dei contenuti mostra le seguenti informazioni:

```
A1 (L) VISICALC
```

Questo comunica che la posizione A1 contiene una label (la L fra parentesi) e specificamente la parola "VISICALC".

Premiamo adesso la freccia a destra per portare il cursore in B1. Il rigo dei contenuti mostra adesso la nuova posizione del cursore, B1, però la posizione A1 del foglio elettronico contiene sempre l'etichetta che abbiamo inserito.

Guardiamo cosa succede quando spostiamo a destra la *finestra*. Premiamo la freccia a destra diverse volte, finchè la colonna A esce dalla finestra: la parola inserita in A1 non è più visibile, però c'è sempre. Per convincercene, usiamo il

comando GO TO per tornare in A1: vi troveremo la parola VISICALC, esattamente come l'avevamo lasciata.

Valori

Portiamo adesso il cursore nella posizione B1 e battiamo un numero, per esempio 1983. Prima di premere il tasto return, si noti il contenuto dei tre righi informativi:

B1
VALUE
1983 □

Il rigo esplicativo comunica che si è battuto un valore, che viene mostrato dal rigo di edit. La posizione B1 ancora non contiene nulla; è necessario premere il tasto return. Premiamolo e guardiamo cosa succede. Il numero compare nella posizione B1 e i righi esplicativo e di edit si svuotano. Il rigo dei contenuti mostra adesso il nuovo valore della posizione B1:

B1 (V) 1983

La V in parentesi informa che B1 contiene un valore.

Ora che sullo schermo ci sono sia un'etichetta che un valore, si può notare un particolare interessante su come il VisiCalc mostra i due diversi tipi di informazione: le etichette vengono automaticamente giustificate a sinistra, mentre i valori a destra. Questo significa che la V della parola VISICALC compare nel primo spazio della posizione A1 e che il 3 di 1983 occupa l'ultimo spazio della posizione B1. Questa forma di rappresentazione è quella normalmente adottata nelle tabelle; di solito le colonne di parole sono allineate sulla sinistra e le colonne di numeri sulla destra.

Riferimenti a valori

Siamo arrivati ai riferimenti a valori, il terzo e più versatile metodo per inserire dati. Un riferimento a valore è un modo per dire al VisiCalc di accedere a un valore di una posizione specifica del foglio elettronico e di riportarlo in un'altra. I riferimenti a valori hanno molteplici applicazioni, come vedremo in questo capitolo e nel successivo. Per adesso ci limiteremo a dare una prima occhiata al loro significato e alla loro utilizzazione.

Spostiamo il cursore in C1, subito a destra del valore appena inserito. Batteremo l'*indirizzo* di quel valore in C1 e vedremo che sarà il valore a esservi trasferito.

Per inserire un riferimento a valore, bisogna prima comunicare al VisiCalc che viene appunto battuto un riferimento a valore, non un'etichetta. Se scrivessimo

"B1" in una posizione, il VisiCalc lo considererebbe etichetta, poichè il primo carattere, la B, non è numerico. Il modo corretto per inserire un riferimento a valore è quello di iniziare l'indirizzo con un +. Oltre alle cifre da 0 a 9, ci sono diversi altri caratteri che il VisiCalc considera contrassegni di inserimenti di dati numerici; fra questi, il segno più, il meno, il punto decimale e la parentesi sinistra.

Quindi proviamo a battere +B1. Sui tre righi superiori dello schermo compaiono le seguenti informazioni:

C1
VALUE
+B1□

Grazie al segno più, il VisiCalc ha effettivamente riconosciuto l'indirizzo B1 come un valore. Premiamo ora il tasto return per vedere cosa succede. Il rigo dei contenuti mostra il riferimento a valore inserito in C1:

C1 (V) +B1

Però la posizione sul foglio elettronico mostra il valore, 1983. In altri termini, inserendo un riferimento a valore nella posizione C1 si è detto al VisiCalc di immettere in C1 lo stesso valore contenuto in B1.

Se adesso si riporta il cursore in B1 e si inserisce un nuovo valore, si vedrà che anche il valore in C1 cambia. Il valore in C1 dipende completamente dal contenuto della posizione B1. Per esempio, se in B1 scriviamo 1984, questo numero compare anche in C1.

EDITING DEI DATI - IL COMANDO /E

Dopo aver visto alcuni degli aspetti fondamentali dell'inserimento dei dati, è arrivato il momento di cominciare a utilizzare i comandi del VisiCalc. Il primo che esamineremo è il comando edit (intervento su dati.)

Si ricorderà che premendo il tasto /, sul rigo di edit compare l'elenco dei comandi. Portiamo il cursore alla posizione A1, che contiene sempre la parola VISICALC, e battiamo /. Guardiamo il rigo dei comandi per vedere se contiene la lettera E, per edit. Se non c'è, la versione del VisiCalc che abbiamo a disposizione non ha la funzione di editing. (Non c'è da preoccuparsi, il comando /E è comodo, ma non essenziale.) Se il rigo dei comandi comprende la E, si batta E sulla tastiera. Ecco cosa compare nei tre righi superiori dello schermo:

A1 (L) VISICALC
(EDIT): LABEL
V ISICALC

Il rigo esplicativo mostra che il VisiCalc è pronto ad accettare modifiche dell'etichetta contenuta nella posizione A1. Il rigo di edit mostra l'etichetta, con una *cue*

che contrassegna la posizione di edit. A questo punto si può scegliere fra diverse azioni:

- si può usare la freccia a destra o quella a sinistra per spostare la *cue* su qualsiasi posizione del rigo di edit;

- si può battere qualsiasi carattere che si desideri *inserire* nel contenuto della posizione (la *cue* indica la posizione in cui viene eventualmente inserito il carattere);

- si può premere il tasto escape per cancellare il carattere subito a sinistra della *cue*;

- si può battere il tasto return per inserire i dati modificati nella posizione corrente;

- si può battere il tasto break per annullare il comando edit; il contenuto originale della posizione rimane immutato.

Per provare il comando edit, cambieremo la parola VISICALC in VISIBILE. Battere otto volte la freccia a destra, per disporre la *cue* alla fine della parola; battere poi quattro volte il tasto escape, per cancellare le ultime quattro lettere della parola. Infine, battere le lettere BILE e poi il tasto return. Il nuovo contenuto di A1 dovrebbe essere la parola VISIBILE.

Può convenire far pratica con il comando edit prima di procedere. Questo comando si dimostrerà utile in seguito, quando cominceremo a inserire formule nel foglio elettronico.

Chi non ha il comando edit non si demoralizzi: si possono ottenere gli stessi risultati semplicemente ribattendo il nuovo contenuto della posizione. Nel caso delle formule, il procedimento sarà più lungo di quanto sarebbe con il comando edit, ma darà lo stesso risultato.

PER CANCELLARE LO SCHERMO E RICOMINCIARE DA CAPO...

Il prossimo comando che studieremo è comodo ma pericoloso: il comando /C restituisce un foglio vergine, uno schermo del VisiCalc vuoto, su cui lavorare. Lo si può usare quando si vuole, per cancellare vecchi dati, formule sbagliate o informazioni che semplicemente non interessano più. Tre rapide battute e si ricomincia da capo. Questo è l'aspetto comodo.

Quello pericoloso è che non rivedremo mai più i dati cancellati, a meno di non averli memorizzati su disco. Finora non abbiamo scritto niente che valga la pena conservare, quindi possiamo essere piuttosto spericolati nell'imparare a usare il comando /C; in seguito, invece, lo si dovrà usare con cautela.

Si batta quindi /C. Si noti che dopo aver battuto /, non è necessario soffermarsi a guardare il rigo dei comandi prima di impartirne uno. Via via che si acquisisce familiarità con le caratteristiche del VisiCalc, diventa sempre meno necessario guardare il rigo dei comandi.

Il comando clear (cancellare) fa comparire sul rigo esplicativo il seguente messaggio:

CLEAR: TYPE Y TO CONFIRM
cancellare: battere Y per confermare

E' la possibilità che viene offerta di cambiare idea: battendo qualsiasi cosa diversa da Y, il comando viene annullato. Ma se si è sicuri di non aver niente da perdere (come adesso), battere Y e guardare cosa succede. Il contenuto di tutto lo schermo scompare per un attimo, poi il VisiCalc ricompare, come se lo si fosse appena lanciato dal disco.

Forse questa è una buona occasione per parlare delle informazioni riguardanti il copyright e la versione. Quando si lancia il programma, o quando si utilizza il comando clear, il rigo esplicativo e quello di edit contengono la scritta di copyright del programma e i numeri che identificano la versione del programma utilizzata. (Quest'ultima informazione può essere necessaria per chiedere informazioni al produttore.) Questi dati possono essere ottenuti in qualsiasi momento, mentre si sta utilizzando il VisiCalc, impartendo il comando versione e cioè battendo semplicemente /V.

Con uno schermo vuoto davanti, stiamo per inserire una colonna di numeri, per preparare alla prossima sezione del capitolo. Useremo la prima colonna di numeri della tabella di cui a Figura 1.1:

11.18
7.62
11.18
1.52
4.83
11.43
6.6
5.59
6.6
6.86

Si noti come quasi tutti i numeri abbiano due cifre decimali e solo un paio una. Questi numeri vanno battuti esattamente in questo modo, perchè utilizzeremo questa differenza per illustrare una possibilità del VisiCalc.

Un'altra utile caratteristica facilita l'immissione di righe o colonne di dati: dopo aver battuto un numero, lo si può inserire nel foglio premendo un tasto direzionale,

invece di return. Questo metodo è più rapido, soprattutto quando i numeri sono molti; premendo un solo tasto si effettuano due operazioni: si inserisce il dato e si sposta il cursore.

Per esempio, scrivere il primo numero dell'elenco, 11.18, nella posizione A1. Premere ora la freccia verso il basso e notare cosa succede. Il valore: 11.18, viene immagazzinato nella posizione A1 e il cursore è nella posizione A2, pronto per l'inserimento del successivo valore della colonna.

Si finisca di inserire i numeri, poi si batta > A1 per riportare il cursore all'inizio. Il foglio elettronico dovrebbe essere come quello della Figura 2.4.

	A
1	11.18
2	7.62
3	11.18
4	1.52
5	4.83
6	11.43
7	6.6
8	5.59
9	6.6
10	6.86

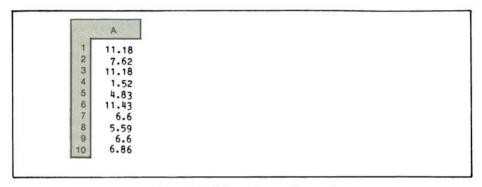



Figura 2.4 - Prima colonna di numeri

CONTROLLO DEL FOGLIO ELETTRONICO

Nel Capitolo 1 abbiamo informalmente suddiviso i comandi del VisiCalc in tre gruppi funzionali: di calcolo, di controllo e di memorizzazione. Stiamo per cominciare a utilizzare alcuni comandi che controllano il foglio elettronico, ma prima vanno discusse alcune caratteristiche della notazione. Nel precedente paragrafo abbiamo detto di battere > A1 invece di "battere il comando GO TO per portare il cursore in A1". Questo è un esempio del tipo di notazione abbreviata con la quale si può descrivere una particolareggiata sequenza di comandi del programma VisiCalc. Non esiste un *metodo standard* per rappresentare tutti i comandi: entro certi limiti, si può creare la notazione che si vuole, purchè il significato di quanto si scrive sia facilmente comprensibile. Ecco alcuni esempi della notazione utilizzata in questo libro:

- /CY (viene impartito il comando clear (cancellare), che va confermato battendo Y)
- /GF$ (viene scelta l'opzione dollari e centesimi di dollaro per il comando formato globale)
- /TV (viene costituita una colonna di titoli fissi).

Per il momento non ci si preoccupi del significato degli ultimi due comandi; li studieremo fra breve. Ci si limiti a considerare i vantaggi di questa notazione. Dopo ogni carattere di un comando multiplo, il VisiCalc mostra le opzioni disponibili sul rigo esplicativo. I principianti tendono a scrivere i comandi lentamente, consultando attentamente il rigo esplicativo a ciascun stadio del procedimento. (In questo capitolo lo faremo anche noi, poichè è la prima volta volta che incontriamo il messaggio dei vari comandi.) Acquisendo esperienza, si esiterà appena nell'impartire un lungo comando come /GF$. Quando si arriva a questo livello, la notazione abbreviata diventa più comprensibile; si comincia a considerare una combinazione di quattro caratteri come quello che davvero è: un unico comando per compiere un'azione specifica.

Torniamo alla nostra colonna di numeri. I primi due comandi che impartiremo sono entrambi globali. Globale significa che il comando ha effetto su tutto il foglio. Battiamo /G per vedere le quattro opzioni del comando globale; questo è quanto appare nel rigo esplicativo:

GLOBAL: C O R F

Le due opzioni centrali, O e R, si riferiscono a calcoli e per adesso le ignoreremo. Le due che ci interessano sono C (per larghezza delle colonne) e F (per formato).

Il comando /GC consente di modificare la larghezza (cioè il numero di spazi) delle colonne del foglio. La larghezza iniziale è di 9 spazi e la si può ridurre fino a un minimo di 3 o aumentare quanto si vuole, fino alla larghezza massima dello schermo. Conviene fissare questa larghezza al minor valore compatibile con le necessità, così da poter vedere sullo schermo quante più colonne possibili.

Per questo esempio, ridurremo la larghezza delle colonne a 7 spazi. Per scegliere l'opzione C del comando globale, basta battere C quando sul rigo esplicativo compaiono le opzioni globali, al che, sempre sul rigo esplicativo, apparirà:

COLUMN WIDTH
larghezza delle colonne

Rispondere premendo 7 e il tasto return: sullo schermo tutte le colonne vengono istantaneamente riorganizzate con una larghezza inferiore e sulla destra dello schermo compariranno una o due colonne in più.

Useremo adesso un comando globale di formato per allineare i punti decimali delle colonne dei numeri: tutti i numeri verranno mostrati con due cifre decimali.

Impartire il comando globale di formato, /GF. Sul rigo esplicativo appariranno le sette opzioni che questo comando offre:

FORMAT: D G I L R $ *

Queste opzioni si riferiscono al modo in cui i dati vengono *mostrati* sullo schermo; i

comandi di formato *non* agiscono sul modo in cui il computer *memorizza* il contenuto delle posizioni. E' una distinzione importante: quello che si *vede* in una posizione può non essere uguale al valore che vi è effettivamente immagazzinato.

Il comando globale di formato, /GF e il comando formato, /F offrono entrambi le stesse opzioni. La differenza consiste nel fatto che /GF agisce su *tutte* le posizioni del foglio elettronico, mentre /F si riferisce solo alla posizione evidenziata dal cursore. Ecco una breve descrizione delle opzioni: I, $ e * fanno sì che i *valori* vengano rispettivamente rappresentati in interi, come dollari e centesimi e in istogrammi. Le opzioni R e L sono rispettivamente per giustificare a destra (*Right*) o a sinistra (*Left*) tanto i valori che le etichette. (Per i valori giustificati a sinistra, ogni valore è preceduto da uno spazio vuoto, affinchè i numeri delle diverse posizioni non vengano scritti uno di seguito all'altro.) L'opzione G riporta i valori e le etichette al formato *generale*, quello in vigore al lancio del VisiCalc, che mostra:

- le etichette giustificate a sinistra, con un massimo di tanti caratteri pari alla larghezza della posizione:

- i numeri giustificati a destra, sempre preceduti da almeno uno spazio vuoto.

Infine, il comando /FD attribuisce a una posizione il formato globale vigente. (Il comando /GFD non ha alcun effetto.)

In questo capitolo e nei seguenti, vedremo molti esempi dei precedenti comandi. L'opzione di formato che adesso ci interessa è quella globale in dollari e centesimi, /GF$: battiamo il tasto e guardiamo cosa succede. Ai due numeri che prima avevano una sola cifra decimale viene aggiunto uno zero, dopo l'ultima cifra, così che i loro punti decimali si allineano con gli altri della colonna. Lo schermo dovrebbe ora essere come quello della Figura 2.5. Si porti il cursore in A9 per esaminare uno dei numeri cambiati: sul foglio elettronico, il numero dovrebbe

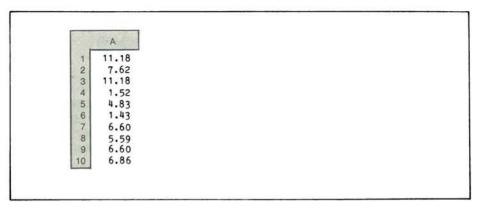



Figura 2.5 - Effetti del comando /GF$

apparire come 6.60. Ma si guardi il rigo dei contenuti, nella parte superiore dello schermo:

A9 (V) 6.6

L'effettivo contenuto della posizione è rimasto immutato; è cambiata solo la sua rappresentazione sul foglio elettronico.

Esaminiamo più dettagliatamente il formato in dollari e centesimi. Si ricordi che questo formato è stato ottenuto tramite un comando *globale*, il che significa che qualsiasi numero inserito sul foglio, comunque lo si scriva, verrà mostrato con due cifre decimali. Proviamo. Spostiamo il cursore in una posizione vuota, sotto la colonna dei numeri, per esempio nella posizione A15. Cominciamo inserendo un numero intero, 2: sul foglio elettronico apparirà come 2.00. Spostiamo adesso il cursore alla posizione A16 e scriviamo il numero 3.246: il numero verrà mostrato come 3.25. Questo mostra un'altra caratteristica del comando /GF$. Numeri con più di due cifre decimali vengono *arrotondati* prima di apparire sullo schermo. Comunque si noti ancora una volta il rigo dei contenuti:

A16 (V) 3.246

Proviamo con un altro numero: ci si porti in A17 e si inserisca 123456. Quanto appare sullo schermo può costituire una sorpresa. Nel formato in dollari e centesimi ci si aspetterebbe che il numero fosse espresso come 123456.00; però ci si deve ricordare che la larghezza delle colonne è di soli 7 spazi, che non sono sufficienti per contenere tale numero. Il VisiCalc comunica che non può mostrare il numero come specificato generando un rigo di simboli "maggiore di":

>>>>>>

Tuttavia il rigo dei contenuti mostra che il numero è stato immagazzinato nella posizione desiderata, anche se non lo si può vedere sul foglio elettronico:

A17 (V) 123456

A questo punto si dovrebbe effettuare anche un altro esperimento. Si è visto cosa succede quando si cerca di inserire un *numero* troppo grande per la larghezza delle colonne del foglio elettronico. Guardiamo cosa succede nel caso di un'etichetta. Si porti il cursore in A18 e si batta la parola VISICALC. Quando abbiamo prima inserito questa parola, le colonne erano larghe 9 spazi, mentre adesso lo sono solo 7. Il rigo dei contenuti mostra sempre tutta la parola, come c'era da aspettarsi:

A18 (L) VISICALC

Però nella posizione sul foglio c'è spazio per 7 caratteri soltanto, quindi ecco cosa si ottiene:

VISICAL

I numeri che abbiamo scritto nelle posizioni A15, A16 e A17 e l'etichetta in A18 servivano solo per fare prove; non vogliamo che rimangano nel foglio elettronico. Come si fa per liberarsene? E' chiaro che non ci si può servire del comando /C, perchè perderemmo anche la colonna di numeri. Useremo invece il comando blank (cancella).

Assicurarsi che il cursore sia in una delle quattro posizioni di cui si vuole cancellare il contenuto, poi battere /B per impartire il comando blank. Sul rigo esplicativo si vedrà semplicemente

BLANK

Se a questo punto viene battuto il tasto return, il valore della posizione corrente scompare. Ripetere l'operazione con gli altri valori.

Battere ora > A1 per riportare il cursore nell'angolo superiore sinistro del foglio elettronico. Siamo pronti per dare un significato ai numeri, e naturalmente questo verrà fatto tramite etichette; inseriremo i nomi dei venditori della Figura 1.3:

BAKER
SMITH
FLINT
BROWN
VERN
MARLOW
HARPER
FLEMING
NASH
WHITE

Vorremmo metterli in una colonna subito a sinistra di quella dei numeri; in altri termini, vorremmo che quella dei nomi fosse la colonna A e che i numeri si spostassero alla colonna B. Purtroppo non abbiamo pensato di lasciare lo spazio per la colonna di etichette..

Il VisiCalc permette di risolvere il problema tramite il comando di inserimento (*insert*). Battere /I: il rigo esplicativo informa della possibilità di inserire una riga o una colonna:

INSERT:R C

In questo caso si vuole una colonna in più, quindi si batterà C. Si tenga d'occhio lo

schermo: la colonna di numeri si sposterà rapidamente a destra di una colonna, lasciando libera la colonna A. Siamo adesso pronti per inserire i nomi dei dieci venditori; si ricordi di usare la freccia verso il basso, non il tasto return, dopo aver battuto ciascun nome. Una volta inseriti tutti i nomi, il foglio elettronico sarà come quello della Figura 2.6.

Quando si cominciano a costruire fogli elettronici grandi, capita spesso di desiderare che una riga o una colonna di etichette rimanga sullo schermo, anche quando si sposta la finestra. Per esempio, supponiamo di avere una tavola di dodici colonne di numeri (come avremo dopo aver inserito i dati dei venditori). Per vedere le ultime colonne sulla destra si deve spostare la finestra, però sarebbe bene che la colonna di etichette rimanesse sullo schermo, per farvi riferimento. Questo è possibile tramite il comando titoli (*titles*).

Si porti il cursore nella posizione A1 e si batta /T per vedere le opzioni che il comando offre:

TITLES: H V B N

	A	B
1	BAKER	11.18
2	SMITH	7.62
3	FLINT	11.18
4	BROWN	1.52
5	VERN	4.83
6	MARLOW	11.43
7	HARPER	6.60
8	FLEMING	5.59
9	NASH	6.60
10	WHITE	6.86

Figura 2.6 - Inserimento nel foglio di una colonna di nomi

H sta per *horizontal* (orizzontale), V per verticale. Le due opzioni consentono rispettivamente di fissare una riga o una colonna di titoli. B, che sta per *both* (entrambe), fissa contemporaneamente sia una riga che una colonna. Per esempio, se il cursore è in A1 e si vuole che rimangano fisse sia la colonna A che la riga 1, basta battere /TB.

In questo esempio, vogliamo che la colonna con i nomi dei venditori rimanga fissa, quindi batteremo V. Si sposti adesso il cursore nella colonna B e si provi poi a riportarlo nella colonna A. Non si può: la colonna è fissata.

Per vedere cosa succede quando la finestra si muove, si sposti il cursore a destra di diverse colonne. Quando la finestra si muove, si perde la colonna B, ma la colonna A rimane sullo schermo.

Esiste un metodo per portare il cursore su una riga o una colonna fisse, se si deve, per esempio, effettuare un cambiamento; va usato il comando GO TO. Si provi, battendo > A1.

Infine, il comando /T offre un'ultima opzione, quella N. Se si vuol eliminare i titoli fissi, si batta /TN. N sta per *none* (nessuno.)

Finora abbiamo sul foglio due colonne contenenti informazioni, una di nomi e una di numeri. Nel Capitolo 1 abbiamo visto che i numeri rappresentano le vendite mensili, espresse in migliaia di dollari, di ciascun venditore. Allo schermo è stato attribuito il formato numerico in dollari e centesimi (/GF$), e abbiamo ridotto la larghezza delle colonne a 7 spazi (/GC7). Abbiamo anche usato il comando *insert* (/I), quello *blank* (/B) e quello titoli (/T). Nelle sezioni successive cominceremo a scrivere formule sul foglio elettronico e poi a replicarle.

SCRITTURA DELLE FORMULE

Nell'esempio dei venditori abbiamo detto che le retribuzioni mensili sarebbero state calcolate a partire da uno stipendio di base di $1000 più una commissione sulle vendite del 5%. Questo lo si può scrivere come

retribuzione = $ 1000 + (.05 X vendite)

Quindi, conoscendo le vendite mensili medie per ogni venditore, possiamo facilmente calcolare le retribuzioni. Per scrivere questa formula sul foglio elettronico del VisiCalc, ci serviremo di quanto abbiamo appreso sui *riferimenti a valori*.

Per cominciare, guardiamo il primo venditore dell'elenco - Baker. Il valore delle vendite per Baker, nella posizione B1, è 11.18, il che significa che in quel mese ha venduto per 11.180 dollari. Per trovare la retribuzione di Baker dobbiamo quindi moltiplicare 11.18 per 0.05 (per trovare la commissione) e aggiungere 1 (che rappresenta lo stipendio di base di $1000.) Questo calcolo verrà effettuato tramite una *formula* del VisiCalc.

Per creare formule per il foglio elettronico, il VisiCalc consente di effettuare le operazioni aritmetiche (addizione, sottrazione, moltiplicazione, divisione ed elevazione a potenza) su riferimenti a valori. I simboli che vengono usati per la scrittura delle formule sono:

+	addizione
—	sottrazione
*	moltiplicazione
/	divisione
^	elevazione a potenza

Quindi, per trovare la retribuzione di Baker, basta scrivere la formula appropriata

usando un riferimento a valore per la posizione B1 che contiene la cifra delle vendite. Gli elementi della formula saranno qualcosa del genere:

1+.05*B1

In altri termini, dobbiamo moltiplicare il contenuto della posizione B1 per .05 e poi aggiungere 1.

C'è però un altro importante aspetto del problema di cui bisogna tenere conto: *l'ordine in cui vengono effettuate le operazioni*. Di solito, nelle espressioni algebriche e nella maggior parte dei linguaggi di programmazione, vige la convenzione standard secondo cui la moltiplicazione e la divisione vengono effettuate *prima* dell'addizione e della sottrazione, indipendentemente dall'ordine in cui le operazioni sono *scritte*. Il risultato, naturalmente, cambia molto nelle formule come la precedente: se 1 viene addizionato a .05 e poi il risultato è moltiplicato per il valore delle vendite, la retribuzione che ne risulta è sbagliata (vantaggiosa per il venditore, non per la società.)

Il VisiCalc, invece, *non* segue la convenzione secondo cui le moltiplicazioni e le divisioni vengono effettuate per prime, ma svolge le operazioni *nell'ordine in cui sono scritte nella formula*. Questo rappresenta un'inevitabile fonte di confusione le prime volte che si usano le formule. Comunque il VisiCalc fornisce un metodo per modificare l'ordine di calcolo: le parentesi. Le espressioni in parentesi vengono calcolate per prime. Per esempio, nella formula della retribuzione, nella quale senz'altro vogliamo che la moltiplicazione venga effettuata prima dell'addizione, si può racchiudere la moltiplicazione fra parentesi, in modo che la formula venga calcolata correttamente:

1+(.05*B1)

Un altro modo di gestire la formula, naturalmente, consiste nell'invertire l'ordine delle operazioni:

.05*B1+1

In questo caso le parentesi non sono necessarie perchè il VisiCalc effettua automaticamente la moltiplicazione per prima, ma è bene abituarsi a racchiudere le operazioni in parentesi, anche quando non è necessario:

(.05*B1)+1

Si eviteranno grossi problemi quando si scriveranno formule lunghe e complicate.

Inseriamo ora la formula e guardiamo quale sarà il risultato. La colonna C sarà

la colonna della retribuzione, quindi spostiamo il cursore in C1. Cominciamo a scrivere la formula:

1+(.05*□

Si noti che il rigo esplicativo mostra VALUE e che la formula compare, carattere dopo carattere, sul rigo di edit. Quando si arriva al punto in cui si deve inserire il riferimento a valore, B1, i metodi possibili sono due. Il primo consiste nel battere normalmente i caratteri e premere il tasto return. Ci sono però molte situazioni in cui tanto le formule che il foglio sono lunghi e complicati, così che non si conosce l'indirizzo della posizione da utilizzare come riferimento a valore. Può darsi che si stia scrivendo una formula e ci si accorga che il valore a cui si vuol far riferimento non è nella finestra. Sarebbe molto scomodo abbandonare la formula in corso di elaborazione, mettersi a cercare nel foglio la posizione del valore al quale si vuol fare riferimento e poi tornare a scrivere la formula.

Fortunatamente il VisiCalc elimina questo inconveniente. Si può spostare il cursore *durante il procedimento dell'immissione di una formula*, purchè si sia arrivati a un punto della formula in cui è appropriato immettere un riferimento a valore. La posizione occupata dal cursore compare sul rigo di edit come riferimento a valore.

Guardiamo come funziona l'operazione:

1+(.05*□

La *cue* indica che il VisiCalc aspetta che la formula venga completata. Il cursore è nella posizione C1; premiamo la freccia a sinistra per portare il cursore in B1 e guardiamo cosa succede sul rigo di edit. Il riferimento a valore, B1, compare automaticamente come parte della formula:

1+(.05*B1□

Poichè si tratta del riferimento a valore desiderato, si può ora chiudere la parentesi per completare la formula. Avvengono due cose. La parentesi compare sul rigo di edit, che quindi mostra:

1+(.05*B1)□

E il cursore torna alla posizione C1.

Premiamo il tasto return per vedere il risultato della formula. La retribuzione di Baker per quel mese è espressa da 1.56 (il che naturalmente significa $1560).

Per riassumere, due sono le cose essenziali da ricordare quando si inseriscono formule che contengono riferimenti a valori. Primo, si può inserire il riferimento a valore battendolo manualmente o portando il cursore alla posizione alla quale si desidera far riferimento. Secondo, il VisiCalc effettua le operazioni nell'ordine in cui sono scritte, a meno che la formula non contenga parentesi.

REPLICA DELLE FORMULE

E' stata così calcolata la retribuzione di Baker, ma ci si deve ancora occupare degli altri venditori. Naturalmente potremmo semplicemente ribattere la formula della retribuzione per ogni venditore. Si otterrebbe il risultato desiderato, ma ci vorrebbe molto tempo e si potrebbero commettere errori.

Il comando di replica (*replicate*), /R, serve esattamente a questo tipo di situazioni: si è scritto una formula e la si vuole *applicare* a molti dati senza doverla riscrivere ogni volta.

Per osservare il comando di replica in funzione, controllare che il cursore sia sempre in C1 e battere /R. Il rigo esplicativo mostrerà il messaggio:

REPLICATE: SOURCE RANGE OR RETURN
replica: campo di origine o return

e sul rigo di edit ci sarà

C1 □

Il comando di replica passa per tre fasi di istruzioni prima di svolgere l'operazione che gli compete. Nella prima, bisogna indicare le formule da replicare, che formano il *source range* (campo di origine). In questo esempio si deve replicare una sola formula, quella che calcola la retribuzione, ma successivamente replicheremo più formule tutte insieme. Il campo di origine del comando di replica permette di far fronte a entrambe le situazioni. Per ora, per indicare che va replicata solo la formula in C1, battiamo semplicemente il tasto return; il rigo di edit mostrerà automaticamente il campo di origine costituito da una sola formula:

C1...C1: □

Il rigo esplicativo mostra un nuovo messaggio:

REPLICATE:TARGET RANGE
replica: campo di destinazione

E' la seconda fase del comando. Bisogna indicare su quale campo dev'essere applicata la formula in C1; tale campo apparirà sul rigo di edit con lo stesso formato del campo di origine:

posizione...posizione

Come nel caso della formula dei riferimenti a valori, si può *battere* gli indirizzi delle due posizioni o spostare il cursore alla posizione che si vuole indicare come parte del campo.

Per completare la colonna C, quindi, con le retribuzioni di ogni venditore, il campo di destinazione sarà costituito dalle posizioni da C2 a C10. Battiamo il primo indirizzo, C2, poi un punto. Sul rigo di edit comparirà quanto segue:

C1...C1: C2... □

Si noti come basti battere il punto una sola volta per ottenerne tre sul rigo di edit. Battiamo adesso C10 e il tasto return. Questo ci farà entrare nella terza, e forse più complessa, fase del comando di replica.

Per effettuare la replica, il VisiCalc deve sapere cosa fare dei riferimenti a valori della formula. Devono essere considerati costanti o variabili? Va usato lo stesso valore ogni volta che viene replicata una formula, oppure bisogna spostarsi nella colonna (o nella riga) e modificare il riferimento a valore a ogni replica?

Guardiamo il rigo esplicativo per vedere qual'è la domanda posta dal VisiCalc:

REPLICATE: N=NO CHANGE, R=RELATIVE
replica: n=nessun cambiamento, r=relativo

Il rigo di edit mostra la formula con la *cue* che indica il riferimento a valore:

C1:C2...C10: 1+(.05* [B1]

Si deve rispondere battendo N o R per ogni riferimento a valore della formula. In questo caso ce n'è uno solo, B1. Si vuole usare B1 (cioè le vendite di Baker) per ogni applicazione della formula della retribuzione, oppure si vuole usare il corrispondente valore della colonna B (B2, B3, B4 e così via)? E' chiaro che il caso in questione è il secondo, quindi si batterà R, per scegliere "relativo".

Poichè questo è l'unico riferimento a valore, si ottiene immediatamente una colonna di numeri che rappresentano le retribuzioni di ogni venditore. Il foglio elettronico dovrebbe essere come quello della Figura 2.7.

	A	B	C
1	BAKER	11.18	1.56
2	SMITH	7.62	1.38
3	FLINT	11.18	1.56
4	BROWN	1.52	1.08
5	VERN	4.83	1.24
6	MARLOW	11.43	1.57
7	HARPER	6.60	1.33
8	FLEMING	5.59	1.28
9	NASH	6.60	1.33
10	WHITE	6.86	1.34

Figura 2.7 - La formula di replica delle retribuzioni

Siamo passati per le tre fasi del comando di replica in un esempio piuttosto semplice. Nella prossima sezione renderemo il foglio elettronico più utile *generalizzando* la formula della retribuzione.

ANCORA SULLA REPLICA

Supponiamo di voler modificare lo stipendio di base e la percentuale delle commissioni dei venditori, perchè riteniamo che la retribuzione attuale non costituisca un incentivo sufficiente per i venditori migliori. Vorremmo disporre di un metodo rapido per esaminare l'effetto sulle retribuzioni lorde di diverse ipotesi di stipendi più commissioni.

Nella precedente versione del foglio elettronico si è scritto lo stipendio di base e le percentuali di commissione direttamente nella formula delle retribuzioni, come costanti. Il metodo ha fornito i risultati corretti, ma in questo caso, in cui si vuole fare esperimenti su altri valori, non è molto comodo.

Un approccio migliore, non limitante, consiste nello scrivere una formula in cui lo stipendio di base e la percentuale di commissione sono *riferimenti a valori* e non costanti. A quel punto è facile modificare i valori per vedere l'effetto dei cambiamenti sulle retribuzioni. Guardiamo il metodo in azione.

Innanzi tutto, si può anche cancellare il contenuto della colonna C, poichè lo si modificherà completamente. Il comando di cancellazione (*delete*) permette di eliminare le informazioni di una colonna o di una riga. Si porti il cursore in C1 e si batta /D; sul rigo esplicativo apparirà:

```
DELETE: R C
```

Battere C per cancellare la colonna.

Adesso vanno specificate due posizioni sulle quali mettere lo stipendio di base e la percentuale delle commissioni. Scegliamo rispettivamente B12 e B13. Cominceremo con la tavola originale delle retribuzioni. Battiamo >B12 e inseriamo il valore 1 (che rappresenta $ 1.000); poi inseriamo .05 (per 5%) in B13.

La nuova formula delle retribuzioni, quindi, conterrà riferimenti a valori in B12 e a B13, al posto delle costanti 1 e .05. Si riporti il cursore in C1 e si scriva la seguente formula:

```
+B12+(B13*B1)
```

(Si ricorda perchè la formula deve cominciare con un segno più?) Si batta return e in C1 dovrebbe comparire il valore 1.56, lo stesso della formula precedente.

Adesso la formula va replicata per tutti i venditori. Alle prime due fasi del comando di replica si dà la stessa risposta di prima: si preme return per indicare il campo di origine e si batte C2.C10 e return per specificare quello di destinazione.

La terza fase richiede un'analisi più attenta. Il salario di base e la percentuale della commissione rimangono invariati per ogni venditore, mentre i valori delle vendite variano ogni volta. Si dovrà quindi battere due N (nessun cambiamento) per B12 e B13 e una R (relativo), per B1. Si noti quanto avviene sul rigo di edit durante questa fase: il VisiCalc dispone la *cue* su un riferimento a valore dopo l'altro e si ferma per permettere di valutare il modo in cui il riferimento va considerato. Ecco come appare la sequenza innanzi tutto quella per B12:

C1: C2...C10:+ [B12]

Si batte N e appare la *cue* per B13:

C1: C2...C10: +B12+ [B13]

Si batte ancora N e compare l'ultima *cue*:

C1: C2...C10: +B12+(B13* [B1]

Infine si batte R, per specificare che il riferimento a B1 va considerato variabile.

Una volta completata la replica, la colonna C del foglio elettronico dovrebbe contenere gli stessi valori di prima; si confronti con la Figura 2.7.

La differenza fra le due versioni della formula è la seguente: adesso si può modificare lo stipendio base e la commissione e conoscere istantaneamente il cambiamento che ne consegue nella colonna delle retribuzioni.

Per esempio, supponiamo di pensare di diminuire lo stipendio base a $500 ma di elevare la commissione al 10%. Basta cambiare i valori in B12 e B13 e guardare i risultati nella colonna C.

	A	B	C
1	BAKER	11.18	1.62
2	SMITH	7.62	1.26
3	FLINT	11.18	1.62
4	BROWN	1.52	0.65
5	VERN	4.83	0.98
6	MARLOW	11.43	1.64
7	HARPER	6.60	1.16
8	FLEMING	5.59	1.06
9	NASH	6.60	1.16
10	WHITE	6.86	1.19
11			
12		0.50	
13		0.10	

Figura 2.8 - La formula generale delle retribuzioni (stipendio base di $500 e commissione del 10%)

Proviamo a farlo. Spostiamo il cursore in B12 e inseriamo il nuovo valore .5 (per $ 500). Adesso inseriamo .1 (per 10%) in B13. Ogni volta che si cambia un valore, il VisiCalc *ricalcola* le formule che vi fanno riferimento, cioè le posizioni da C1 a C10. Il nuovo foglio elettronico (con $ 500 e 10%) dovrebbe essere come quello della Figura 2.8. Si noti il risultato della nuova tavola delle retribuzioni: i venditori migliori vedrebbero aumentata la propria retribuzione, mentre quella dei venditori peggiori risulterebbe ridotta.

Nel Capitolo 5 analizzeremo ancora questo metodo per rendere le formule del VisiCalc più generali. E' a questo che si riferiscono gli utenti del VisiCalc quando parlano di vedere "cosa succederebbe se". Cosa succederebbe se si cambiasse la tabella delle retribuzioni? E se un venditore riesce ad aumentare le proprie vendite del 50%? Con il VisiCalc è facile e rapido ottenere risposta a queste domande.

FINESTRE E GRAFICA

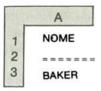
Nei prossimi capitoli amplieremo notevolmente il foglio elettronico, ma prima di passare oltre, abbelliamo un po' la versione attuale. Specificatamente, in questa sezione aggiungeremo alcune intestazioni delle colonne, divideremo la finestra e costruiremo un istogramma.

Cominciamo a creare, tramite il comando di inserimento, due righe vuote nella parte superiore del foglio elettronico. Disponiamo il cursore sulla riga 1 e battiamo due volte /IR. La tabella delle vendite si sposterà in basso, andando a occupare le righe da 3 a 12. Portiamo adesso il cursore in A1. (Si dovrà usare il comando GO TO; ricordiamo perchè?) inseriamo l'etichetta NOME; poi scriviamo VEND. e RETRIB. rispettivamente in B1 e C1. Affinchè queste due ultime etichette corrispondano più chiaramente alle colonne di numeri, giustificati a destra, cui si riferiscono, può convenire riformattarle, cioè attribuire loro un nuovo formato. Un'etichetta può essere giustificata a destra tramite il comando /FR: basta impartirlo quando il cursore è nella posizione su cui si deve operare.

Si sposti adesso il cursore in A2 (usando nuovamente il comando GO TO, dato che la colonna A è fissa come colonna di titoli.) Tracceremo una riga doppia (in effetti formata da segni uguale) fra le intestazioni delle colonne e la tabella vera e propria. A questo scopo si usa il comando di ripetizione delle etichette, /-. Quando lo si impartisce, sul rigo esplicativo compare il seguente messaggio:

LABEL:REPEATING
etichetta:ripetizione

Si può inserire qualsiasi carattere o serie di caratteri che si vuol far ripetere nella posizione. Battere il segno uguale (=) e premere return: nella posizione A2 compariranno sette segni uguale:

	A
1	NOME
2	=======
3	BAKER

Si noti come il rigo dei contenuti descrive la posizione:

A2(/—) =

Se in seguito si decidesse di aumentare o diminuire la larghezza delle colonne, il carattere da ripetere (in questo caso =) continuerà a occupare tutta la posizione.

Un'etichetta da ripetere può essere replicata. Mettendo il cursore in A2, si replichi l'etichetta, tramite il comando di replica /R, nelle posizioni B2 e C2. La Figura 2.9 mostra quale dovrebbe essere l'aspetto del foglio elettronico.

Adesso aggiungeremo allo schermo del VisiCalc una seconda finestra. I motivi per cui può convenire avere due finestre sono svariati; uno, al quale abbiamo già accennato, è per vedere contemporaneamente due parti distanti del foglio. Un altro, forse meno evidente, è di creare due versioni del foglio elettronico con *formato diverso*. Quando si hanno due finestre, si può assegnare a ognuna formati globali, titoli fissi e larghezza delle colonne diversi, ed è proprio quello che faremo adesso.

	A	B	C
1	NOME	VEND.	RETRIB.
2	=======	=======	=======
3	BAKER	11.18	1.62
4	SMITH	7.62	1.26
5	FLINT	11.18	1.62
6	BROWN	1.52	0.65
7	VERN	4.83	0.98
8	MARLOW	11.43	1.64
9	HARPER	6.60	1.16
10	FLEMING	5.59	1.06
11	NASH	6.60	1.16
12	WHITE	6.86	1.19
13			
14		0.50	
15		0.10	

Figura 2.9 - Tavola delle vendite con intestazioni delle colonne

Portare il cursore in D1 e battere il comando finestra /W. Ecco cosa compare sul rigo esplicativo:

WINDOW:H V 1 S U
finestra: H V 1 S U

H e V indicano rispettivamente la suddivisione orizzontale (*Horizontal*) e Verticale. Si può suddividere lo schermo in entrambi i sensi. L'opzione 1 è per tornare a una sola finestra. (Le opzioni S e U sono per ottenere lo spostamento Sincronizzato e non sincronizzato (*Unsynchronized*) delle due finestre.)

Vogliamo suddividere verticalmente il foglio alla colonna D, quindi battiamo V: adesso c'è una finestra a destra e una a sinistra. Il cursore può essere solo in una finestra alla volta; per spostare il cursore da una finestra all'altra, battere il punto e virgola. Si provi a farlo diverse volte, per vedere come funziona il procedimento, poi si metta il cursore nella nuova finestra sulla destra.

Quando si crea una nuova finestra, questa assume un formato con tutte le caratteristiche della finestra singola originale. Si noti come la finestra a destra abbia colonne larghe 7 caratteri, una colonna di titoli fissa (la colonna A) e il formato globale in dollari e centesimi. Vogliamo adesso cambiare tutti questi parametri.

Innanzi tutto, per eliminare i titoli fissi, battiamo /TN (per *niente titoli*). Poi aumentiamo la larghezza delle colonne della finestra sulla sinistra battendo /GC13. Si noti come nessuna di queste due modifiche ha effetto sulla finestra a sinistra.

Portiamo ora il cursore in B1 della finestra di destra: si vedrà la stessa colonna di valori delle vendite a sinistra, ma giustificati a destra in una colonna molto più larga. Per fornire una rappresentazione visiva dei valori delle vendite, potrebbe essere utile costruire un istogramma. E' degno di nota che lo si possa fare impartendo un solo comando; l'opzione * del comando di formato globale sostituisce i numeri con una fila di asterischi. Per esempio, il numero 10 si trasforma in dieci asterischi; il comando non ha alcun effetto sulle etichette.

Ci si prepari a una piacevole sorpresa quando si impartirà il comando /GF*: la colonna B si trasforma istantaneamente in un istogramma delle vendite. Si capisce

	A	B	C
1	NOME	VEND.	RETRIB.
2	=======	=======	=======
3	BAKER	11.18	1.62
4	SMITH	7.62	1.26
5	FLINT	11.18	1.62
6	BROWN	1.52	0.65
7	VERN	4.83	0.98
8	MARLOW	11.43	1.64
9	HARPER	6.60	1.16
10	FLEMING	5.59	1.06
11	NASH	6.60	1.16
12	WHITE	6.86	1.19
13			
14		0.50	
15		0.10	

	B
1	VEND.
2	=============
3	***********
4	*******
5	***********
6	*
7	****
8	***********
9	******
10	*****
11	******
12	******
13	
14	
15	

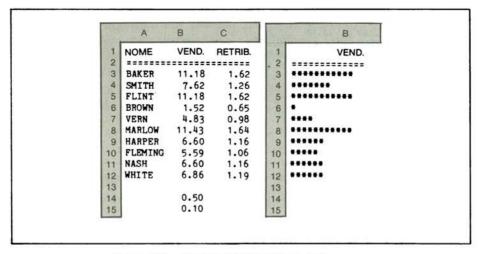



Figura 2.10 - Una seconda finestra per l'istogramma

adesso perchè abbiamo aumentato la larghezza delle colonne; per farvi entrare tutti gli asterischi dell'istogramma. Il foglio elettronico dovrebbe essere come quello della Figura 2.10.

MEMORIZZAZIONE DEI FOGLI ELETTRONICI - /SS E /SL

Dopo aver molto lavorato per costruire un foglio elettronico, lo si vorrà naturalmente memorizzare su disco, così da poterlo successivamente richiamare o modificare. Questo viene effettuato tramite il comando di memorizzazione (*storage*) /S. Si può anche caricare (*load*) nuovamente il file del VisiCalc nella memoria del computer; cancellare (*delete*) un file dal disco; inizializzare (*initialize*) un disco per poterlo usare per la memorizzazione; abbandonare (*quit*) il VisiCalc e memorizzare un file DIF - il tutto tramite il comando /S.

Impartiamo il comando e guardiamo le opzioni che corrispondono alle precedenti funzioni:

STORAGE:L S D I Q #

Controllare di avere nel drive un disco inizializzato in modo corretto. (Se necessario, inizializzare un disco nuovo con il comando /SI.) Scegliere l'opzione S del comando per memorizzare il foglio. Il rigo esplicativo mostrerà il seguente messaggio:

STORAGE: FILE FOR SAVING
memorizzazione: file in cui memorizzare

e comparirà la *cue*, in attesa che venga battuto un nome di file. Possiamo decidere di chiamare questo file VENDITE.

Dato che il VisiCalc permette di memorizzare su disco più di un tipo di file, conviene aggiungere un'estensione al nome di un file. Per esempio, VENDITE.VC potrebbe indicare un file contenente un foglio elettronico del VisiCalc, mentre VENDITE.DIF un file DIF. (Alcune versioni del programma includono automaticamente queste estensioni.)

Una volta battuto un nome di file, premere il tasto return e il file verrà memorizzato. Si può ora cancellare il foglio (/CY) e cercare di *ricaricare* il file dal disco nel VisiCalc. Battere di nuovo /SL e di nuovo il rigo esplicativo chiede un nome di file, però questa volta sarà quello del file che si vuol caricare:

STORAGE: FILE TO LOAD
memorizzazione: file da caricare

Si può semplicemente battere il nome del file, VENDITE.VC, e premere return. Ma cosa succede se non si ricorda il nome di tutti i file del disco? Come si può ottenere

un elenco di tali nomi? Basta premere la freccia verso destra invece di battere un nome: il VisiCalc accederà al disco e mostrerà (sul rigo di edit) il nome del primo file. Per esempio:

```
STORAGE:FILE TO LOAD
VENDITE.VC□
```

Si noti la *cue* che aspetta la risposta. Se a questo punto si batte return, il VisiCalc carica il file VENDITE.VC. In seguito, dopo aver memorizzato sul disco diversi file, la freccia a destra permette di mostrare il nome di tutti quanti, uno alla volta. In questo modo si può esaminare tutto l'elenco di file su disco (chiamato anche *catalogo* del disco), oppure si può decidere di caricare un file premendo return.

Carichiamo VENDITE.VC per verificare che l'operazione sia stata effettuata in modo corretto. Il foglio elettronico dovrebbe avere lo stesso aspetto di prima (vedi Figura 2.10).

SOMMARIO

I tre tipi di inserimento di dati nel programma VisiCalc generano dati di due tipi: i valori sono dati numerici, le etichette dati non numerici. I riferimenti a valori, che sono gli elementi costitutivi delle formule, generano valori numerici.

Abbiamo suddiviso i comandi del VisiCalc in tre categorie. I comandi di controllo esaminati finora comprendono:

/B	(per cancellare il contenuto di una posizione)
/CY	(per cancellare tutto il foglio)
/D	(per cancellare una riga o una colonna)
/E	(per intervenire sul contenuto di una posizione)
/GC	(per modificare la larghezza delle colonne)
/GF$	(per attribuire il formato in dollari e centesimi a dati numerici)
/FG*	(per creare un diagramma grafico dai dati numerici)
/I	(per inserire una riga o una colonna)
/T	(per fissare i titoli)
/W	(per creare una seconda finestra)

I comandi di formattazione agiscono solo sull'*aspetto* dei dati, non sull'effettivo contenuto della memoria. Le due finestre possono essere utilizzate per vedere due diverse parti del foglio elettronico o per vederne una in due formati diversi.

Il comando di replica applica una formula a un'ampia gamma di posizioni. Tale comando conduce attraverso tre fasi: va specificato il campo di origine delle formule e il campo di destinazione; infine, va indicato per ciascun riferimento a valore della formula se vada inteso come costante (*no change*) o variabile (*relative*).

E' spesso possibile generalizzare le formule, e vedere "cosa succederebbe se", scrivendole in termini di riferimenti a valori piuttosto che con numeri espliciti.

Il versatile comando /S fornisce diverse possibilità di memorizzare file su disco e di ricaricarli sul foglio.

CAPITOLO 3

LE FUNZIONI DEL VISICALC (PARTE I)

COME UTILIZZARE LE FUNZIONI

Per rendere la scrittura delle formule più semplice, più efficace e più efficiente, il VisiCalc offre una quantità di *funzioni* che possono, appunto, essere incluse nelle formule del foglio elettronico. L'uso di alcune formule è semplice, anche se si tratta di funzioni importanti, con un'ampia gamma di applicazioni come, per esempio, quelle della somma e della media. Altre, come la funzione del valore attuale netto e quelle trigonometriche ed esponenziali, sono invece per un'utilizzazione più specialistica e possono richiedere maggiori nozioni matematiche. La scelta delle funzioni da imparare dipenderà dalle necessità e dagli scopi particolari per cui verrà utilizzato il VisiCalc.

Le funzioni del VisiCalc possono essere suddivise in categorie. Le semplici funzioni aritmetiche comprendono la somma, la media, il valore assoluto, il valore intero, il conto (per trovare il numero di valori di un elenco), il massimo e il minimo. Le funzioni matematiche più avanzate sono la radice quadrata, i logaritmi naturali e quelli in base 10, l'elevazione a potenza dell'esponente naturale e l'insieme delle funzioni trigonometriche. In questo capitolo esamineremo con attenzione le funzioni aritmetiche, mentre tratteremo più superficialmente quelle matematiche più avanzate; nei Capitoli 4 e 5 considereremo le altre categorie di funzioni.

Tutte le funzioni hanno un aspetto in comune: è necessario battere un carattere speciale per informare il VisiCalc che verrà utilizzata una funzione. Questo carattere è il simbolo "@", che deve essere il primo di tutte le funzioni. Per esempio:

```
@SUM
@AVERAGE
@SIN
@MAX
```

In questo contesto, il segno @ non ha un significato intrinseco, ma serve solo per comunicare al VisiCalc che la parola che segue è il nome di una funzione.

Inoltre, la maggior parte delle funzioni del VisiCalc (ma non tutte) richiedono che vengano specificati i valori su cui devono operare. Per esempio, se si usa la

funzione della media, si devono indicare i valori di cui dev'essere calcolata la media. Per usare la funzione della radice quadrata, invece, basta specificare un solo valore. Tali valori costituiscono l'*argomento* della funzione. Funzioni diverse richiedono argomenti di tipo diverso; la maggior parte delle funzioni che esamineremo in questo capitolo richiedono un argomento costituito da un solo valore o da un elenco di valori.

L'argomento di una funzione del VisiCalc viene scritto fra parentesi, subito dopo il nome della funzione. Gli argomenti possono essere espressi in svariati modi. Talvolta saranno semplicemente dei numeri, come:

```
@SUM(5,7.2,8.34,16)
```

(Si noti come tutti gli elementi dell'elenco siano separati gli uni dagli altri da una virgola.) Più comunemente, invece, gli argomenti delle funzioni sono costituiti da riferimenti a valori. Per esempio, si può specificare un *campo* di posizioni:

```
@SUM(B1...B12)
```

o un *elenco* di posizioni:

```
@SUM(A3, B15,D20)
```

Nell'argomento di una funzione possono anche essere inserite *altre funzioni*:

```
@SUM@(SQRT(B5),@MIN(A3...A10))
```

Questa formula trova la somma di due valori: la radice quadrata del valore in B5 e il più piccolo dei valori nel campo di posizioni da A3 ad A10. In questo capitolo vedremo alcuni esempi realistici sull'uso di funzioni come argomenti.

Infine, in certi casi, l'argomento sarà costituito da una *combinazione* di numeri, riferimenti a valori e altre funzioni. Per esempio, la formula

```
@SUM(135.22,C3...H3,@MAX(A1...A5))
```

trova la somma di un elenco di valori; l'elenco è costituito da un numero (135.22), un campo di riferimenti a valori (C3...H3) e il valore massimo di un campo (@MAX(A1...A5)).

In questo capitolo e nei due successivi esamineremo molte delle funzioni del VisiCalc e vedremo esempi di come utilizzarle. In molti esempi, cominceremo a costruire i fogli completi introdotti nel Capitolo 1. Si ricorderà che questi tre fogli elettronici, che sono stati introdotti per illustrare le possibilità del VisiCalc, usavano tutti gli stessi valori numerici (vedi Figura 1.1). E' arrivato il momento di inserire quella tavola in un foglio elettronico del VisiCalc. Poichè utilizzeremo quei valori in più contesti, conviene memorizzarli in un file su disco in formato nudo (cioè senza

(cioè senza etichette o formule). Una volta inseriti tutti i numeri, è bene controllare di non aver commesso errori di battitura. Si usi il comando di formato globale /GF$ per attribuire a tutti i valori il formato in dollari e centesimi.

Verificare che nel drive ci sia un disco per i dati e battere il comando /SS per memorizzare il foglio. Attribuire al file il nome NUMERI.VC. Il foglio elettronico verrà memorizzato su disco con l'esatto formato che ha sullo schermo.

Il primo esempio completo che svolgeremo è quello sull'applicazione di economia domestica: quello in cui si annotano le spese per il cibo delle ultime dodici settimane. Prima di cominciare conviene inserire etichette per le righe e le colonne delle tavole di numeri. Portiamo il cursore in A1 e utilizziamo il comando di inserimento, /IC, per creare una prima colonna vuota. Poi, cominciando da A1, battiamo le categorie di spesa:

Si sono incontrate difficoltà con l'ultima etichetta, "**TOTALI"? La fonte dei problemi è la seguente: l'asterisco (*) viene abitualmente riservato per le formule (per indicare la moltiplicazione) e il VisiCalc non sa come gestirlo quando è il primo carattere di un inserimento di dati. E' necessario rassicurare il VisiCalc informandolo che si sta inserendo un'etichetta che casualmente inizia con "*". Il VisiCalc consente un piccolo trucco per inserire come etichette caratteri normalmente associati a valori. Questo trucco consiste nel battere virgolette invisibili come primo carattere dell'inserimento. Proviamo: in A11 battiamo le virgolette ("), e sui primi tre righi dello schermo si vedrà:

A11
LABEL
□

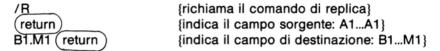
Il rigo esplicativo mostra che il VisiCalc è pronto per ricevere un'etichetta, mentre il rigo di edit contiene solo la *cue*. Adesso si possono battere i due asterischi e la parola TOTALI.

Questa tecnica può essere utilizzata tutte le volte che si vuole inserire un numero (o un simbolo come +, —, =, /, @) che deve essere interpretato come etichetta.

Dopo aver inserito tutte le etichette nella colonna A, facciamo spazio nella parte superiore del foglio elettronico per inserirvi le intestazioni delle colonne. Andiamo in A1 e battiamo due volte /IR. (Si avrà così una riga per le intestazioni e una da tenere vuota.) E' meglio se le intestazioni delle colonne sono giustificate a destra, perchè sia più facile associarle con le colonne di numeri alle quali si riferiscono. Nella posizione B1, battiamo /FR. Il rigo dei contenuti mostrerà che la posizione è formattata e che non contiene nulla:

B1/FR

Si può replicare il formato su tutta la riga vuota (da B2 a M2), in modo che ogni intestazione venga giustificata a destra non appena viene battuta. Impartire la seguente sequenza di comandi:

/R {richiama il comando di replica}
(return) {indica il campo sorgente: A1...A1}
B1.M1 (return) {indica il campo di destinazione: B1...M1}

In tutto il resto del libro ci saranno "elenchi" di comandi del VisiCalc nel formato di questo esempio. Sulla sinistra appaiono gli effettivi comandi che batteremo; sulla destra (in parentesi) i commenti di spiegazione della funzione dei comandi. (Naturalmente i commenti sono inclusi solo per chiarezza; non fanno parte del foglio elettronico.) Il formato dell'elenco dei comandi non ha niente di *standard*; quando si costruisce un foglio elettronico se ne può scegliere un altro. Il suo solo requisito è che i comandi dovrebbero essere chiari non solo a chi li impartisce, ma anche alle altre persone che possono aver a che fare con essi.

Inserire nella riga 1 le intestazioni delle colonne, da "SETT.#1" a "SETT.#12", poi andare in A1 e battere /TB per fissare sia i titoli orizzontali che quelli verticali. Il foglio elettronico dovrebbe essere come quello della Figura 3.1.

Siamo pronti per cominciare ad applicare le funzioni del VisiCalc a questo foglio elettronico.

LE FUNZIONI ARITMETICHE

I dati sulla spesa per il cibo sono nelle colonne da B a M incluse. Creeremo una riga dei totali nella 13 (cioè, dei totali settimanali) e una colonna dei totali nella N (cioè, dei totali di ogni categoria). Poi nelle colonne da O a R creeremo altre colonne di dati statistici che descriveranno il comportamento di chi fa la spesa. Le funzioni che verranno utilizzate per creare queste colonne sono riassunte nella Figura 3.2.

	A	B	C	D	E	F
1		SETT. #1	SETT. #2	SETT. #3	SETT. #4	SETT. #5
2						
3	MISC	11.18	6.35	8.13	3.56	2.54
4	CEREALI	7.62	7.11	5.59	3.30	0.25
5	LATTERIA	11.18	7.62	6.35	4.06	1.02
6	GATTO	1.52	1.78	3.05	4.83	6.86
7	VARIE	4.83	5.33	7.62	9.91	9.91
8	VEGET.	11.43	11.68	13.97	10.67	10.67
9	SIGAR.	6.60	5.59	7.87	8.89	8.89
10	CARNE	5.59	5.08	5.33	9.30	15.49
11	DOLCI	6.60	6.35	8.38	7.37	9.55
12	LIQUORI	6.86	7.37	9.40	8.38	8.89
13	**TOTALI					

	G	H	I	J	K	L	M
1	SETT. #6	SETT. #7	SETT. #8	SETT. #9	SETT. #10	SETT. #11	SETT. #12
2							
3	0.76	1.52	2.03	1.78	3.81	7.62	9.40
4	0.00	0.00	0.00	0.51	0.76	5.08	5.59
5	0.25	0.00	0.00	0.51	2.54	5.84	10.16
6	4.83	4.57	3.30	2.79	2.79	2.03	1.02
7	11.18	9.40	7.37	7.37	7.11	6.35	5.08
8	11.94	17.02	13.46	14.22	5.84	9.91	12.95
9	8.38	3.89	7.87	7.11	6.60	7.11	6.10
10	22.86	17.53	17.24	22.10	20.83	6.86	4.06
11	8.89	10.41	12.09	7.87	6.86	7.37	7.62
12	7.62	9.40	10.16	8.38	7.37	9.65	8.89
13							

Figura 3.1 - Dodici settimane di spese domestiche (vedi Figura 1.2)

Cominceremo trovando le spese totali di ogni settimana. Portiamo il cursore in B13, cioè alla fine della colonna relativa alla prima settimana. Battiamo il carattere distintivo delle funzioni ,@, e notiamo il messaggio nella parte superiore dello schermo:

B13
VALUE
@□

Si può vedere come il VisiCalc abbia considerato il simbolo @ come l'inizio di un valore. Si batta ora il nome della funzione, SUM (somma), e l'argomento, (B3.B12). Al solito, si possono inserire i due indirizzi battendoli carattere per carattere sul rigo di edit, o spostando il cursore nella posizione alla quale si vuol fare riferimento

Funzione	Argomento	Descrizione
@SUM	elenco	Trova la somma dei valori dell'elenco
@AVERAGE	elenco	Trova la media dei valori dell'elenco
@COUNT	elenco	Trova il numero di posizioni non vuote dell'elenco. (Non conta le etichette.)
@INT	valore	Trasforma il valore in numero intero (troncando, non arrotondando.)
@MAX,	elenco	Trova il valore massimo o minimo
@MIN	elenco	dell'elenco
@ABS	valore	Trasforma il valore (positivo o negativo) nel corrispondente valore positivo (valore assoluto).

Figura 3.2 - Le funzioni aritmetiche

nella formula. Vogliamo sommare i valori delle posizioni da B3 a B12; sul rigo di edit dovrebbe esserci:

@SUM(B3...B12)□

Inserita la funzione, battiamo il tasto return e guardiamo il risultato. La spesa totale per la settimana # 1 è stata di $73.41.

Le funzioni possono essere replicate come qualsiasi altra formula. A titolo di esempio, battiamo quanto segue (il cursore dovrebbe essere in B13):

/R	{richiama il comando di replica}
(return)	{indica il campo di origine: B13...B13}
C13.M13 (return)	{indica il campo di destinazione: C13...M13}
RR	{entrambi i riferimenti a valori sono relativi}

Si noti che i riferimenti a valori nella formula SUM, B3 e B12, vanno replicati tutti e due come *relativi*.

La riga dei totali settimanali comparirà sotto le colonne; spostiamo il cursore sulla riga per esaminare i singoli totali:

SETT. #1	SETT. #2	SETT. #3	SETT. #4	SETT. #5	SETT. #6
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
73.41	64.26	75.69	70.27	74.07	76.71
SETT. #7	SETT. #8	SETT. #9	SETT. #10	SETT. #11	SETT. #12
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
73.74	73.52	72.64	64.51	67.82	70.87

Si noti il rigo dei contenuti mentre il cursore si sposta sulla riga: i riferimenti a valori cambiano a ogni immissione. Per esempio, per F13, che contiene il totale della settimana 5, il rigo dei contenuti mostra:

```
F13(V) @SUM(F3...F12)
```

Creeremo adesso una colonna dei totali di ogni categoria di spesa. Portiamo il cursore in N1 e battiamo un'etichetta di colonna:

>N1	{sposta il cursore}
/FR	{giustificazione a destra per la posizione}
TOTALI (return)	{inserimento dell'etichetta}

Spostiamo adesso il cursore in N3 e inseriamo la formula dell'addizione:

>N3	{sposta il cursore}
@SUM(B3.M3) (return)	{inserimento della formula}

La somma 58.68 dovrebbe apparire in N3. Adesso si può replicare la formula su tutta la colonna:

/R	{richiama il comando di replica}
(return)	{indica il campo di origine: N3...N3}
N4.N13	{indica il campo di destinazione: N4...N13}
RR	{entrambi i riferimenti a valori sono relativi}

Ecco come dovrebbe apparire adesso la colonna N:

	A	N
1		TOTALI
2		
3	MISC.	58.68
4	CEREALI	35.81
5	LATTERIA	49.53
6	GATTO	39.37
7	VARIE	91.46
8	VEGET.	143.76
9	SIGAR.	84.90
10	CARNE	152.27
11	DOLCI	99.36
12	LIQUORI	102.37
13	**TOTALI	857.51

Si noti che la posizione N13 mostra una cifra per la spesa totale del periodo: $ 857.51.

Finora abbiamo visto che le funzioni possono essere trattate come qualunque altro tipo di formula: il replicarle è altrettanto facile e forse addirittura più efficace, perlomeno in termini di risultati potenziali.

Per illustrare qualche altra funzione del VisiCalc, aggiungeremo al foglio elettronico quattro colonne contenenti dati statistici. La colonna O mostrerà la somma massima spesa in una settimana per ogni categoria; la colonna P mostrerà quella minima. La colonna Q conterrà la spesa media di ogni categoria; infine, la colonna R mostrerà le percentuali delle spese di ogni categoria rispetto alla spesa totale del periodo ($ 857.51). Forse il foglio elettronico darà più informazioni di quante se ne desiderava, però c'è un punto che dovrebbe ormai essere chiaro: l'efficacia e la versatilità del VisiCalc risiedono nella memorizzazione e nella manipolazione dei numeri; spetta all'utente decidere quali dati generare, e come utilizzare tutti i numeri di una data applicazione.

Nel creare le colonne addizionali, vedremo un nuovo aspetto del comando di replica: invece di replicare le formule di ogni colonna una per una, le replicheremo tutte insieme, tramite un'efficiente operazione. Il primo passaggio sarà di costruire le formule e di inserirle nel foglio elettronico per la prima categoria di spesa (da O3 a R3), poi estenderemo le formule a tutti i dati per le quattro nuove colonne. Portiamo il cursore in O1 e determiniamo nel seguente modo il formato delle quattro posizioni da O1 a R1:

```
>O1                   {sposta il cursore}
/FR                   {giustificazione a destra}
/R                    {richiama il comando di replica}
(return)              {indica il campo di origine: O1...O1}
P1.R1 (return)        {indica il campo di destinazione: P1...R1}
```

Inseriamo poi le intestazioni, "MAX", "MIN", "MEDIA", "PERCENT" nelle quattro posizioni.

La prima formula da inserire, nella posizione O3, è quella del valore massimo:

```
>O3 @MAX(B3.M3)
```

(La notazione >O3 in questo caso non implica che si debba usare il comando GO TO per portare il cursore in O3; significa semplicemente "questa è la formula da inserire in O3." Si ricordi anche che per inserire la formula è necessarrio battere il tasto return.)

Nella posizione P3 viene inserito il valore minimo:

```
>P3 @MIN(B3.M3)
```

e in Q3 il valore medio:

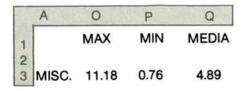
```
>Q3 @AVE(B3.M3)
```

Mentre si inserisce quest'ultima formula, si guardi il rigo di edit. Quando si apre la parentesi, il VisiCalc attribuisce automaticamente alla funzione il suo nome per intero:

```
@AVERAGE(□
```

Nella maggior parte delle versioni del VisiCalc, questa possibilità può essere utilizzata per inserire il nome di ogni funzione. Non appena si sono battute abbastanza lettere per identificare una funzione (così che sia chiaramente distinta da tutte le altre) si può aprire la parentesi e il VisiCalc completa il nome. (Alcune versioni del programma richiedono invece che il nome venga battuto per intero.)

I primi tre inserimenti di funzione dovrebbero avere questo aspetto sul foglio elettronico

	A	O	P	Q
1		MAX	MIN	MEDIA
2				
3	MISC.	11.18	0.76	4.89

L'ultima formula, per la percentuale delle spese totali, sarà un po' più complicata. La esamineremo passaggio per passaggio, cominciando dalla formattazione dei dati. Il formato globale del foglio elettronico è in dollari e centesimi; supponiamo che si preferisca esprimere le percentuali con un numero a una sola cifra decimale, arrotondata. In altri termini, se una percentuale è

4.59120

la si vuole esprimere come

4.6

Il VisiCalc non dispone di un comando di formattazione che esegua automaticamente questa operazione, ma si può scrivere una formula di "arrotondamento" tramite la funzione @INT. La prima volta che la si utilizza si può rimanere un pò confusi, ma con la pratica si riuscirà ad adattarla a ogni situazione.

Il procedimento di arrotondamento si effettua in quattro passaggi:

1. Moltiplicare il numero per una potenza di 10, in modo che il punto decimale compaia subito dopo la cifra che si vuole arrotondare, in eccesso o in difetto.

2. Aggiungere al numero .5.

3. Trovare il valore intero del risultato (tramite la funzione INT.)

4. Dividere il numero per la stessa potenza di 10 del passaggio 1.

Ecco quale sarebbe il procedimento per arrotondare al decimo il numero 4.59120:

1. 4.59120 x 10 = 45.9120
2. 45.9120 + .5 = 46.4120
3. @INT(46.4120) = 46
4. 46/10 = 4.6

I quattro passaggi possono essere espressi tramite una sola formula del VisiCalc. Supponiamo che la posizione A1 contenga un numero che si vuole arrotondare al decimo: porteremo il cursore nella posizione in cui dovrà andare il valore arrotondato e scriveremo

@INT(10*A1+.5)/10

Essenzialmente, questa è la formula di arrotondamento che utilizzeremo per la colonna delle percentuali; soltanto che il riferimento a un solo valore (il precedente

A1) sarà sostituito da una breve formula per calcolare la percentuale. Per la prima categoria di spesa, questa formula sarà

100*N3/N13

poichè la posizione N13 contiene le spese totali di tutto il periodo. (Moltiplicando la frazione per 100 la si trasforma da un numero decimale in una percentuale.) Quando questa formula viene combinata con quella di arrotondamento, otteniamo:

@INT(1000*N3/N13+.5)/10

Prima di inserire questa formula nella posizione R3, sarà necessario riformattare la posizione. Vogliamo che questa colonna abbia un formato diverso da quello del resto del foglio elettronico; in effetti, vorremmo tornare al formato numerico *generale*. Per far questo, si porti il cursore in R3 e si batta semplicemente /FG. Quando la formula viene replicata sulla colonna R, il formato si riferirà a tutte le sue posizioni.

Battiamo ora la formula ed esaminiamo i risultati che produce. La categoria di spesa "miscellanea" rappresenta il 6.8% delle spese totali del periodo di dodici settimane.

Siamo adesso pronti a replicare le quattro formule sulle colonne corrispondenti per completare la nostra opera su questo foglio elettronico. Portiamo il cursore in O3 e battiamo /R. Il rigo esplicativo mostra:

REPLICATE: SOURCE RANGE OR RETURN
replica: campo di origine o return

Finora, a questo punto, abbiamo sempre battuto il tasto return per comunicare al VisiCalc che dovevamo replicare solo una formula. Questa volta abbiamo le quattro formule, quindi il procedimento di replica sarà un po' diverso. Battiamo:

.R3 (return)

per indicare il campo di origine. Il rigo di edit dovrebbe essere il seguente:

O3...R3: □

Per il campo di destinazione, battere:

O4.O13 (return)

Adesso, nella terza fase del comando di replica, il VisiCalc farà considerare le quattro formule di origine una per una perchè venga specificato come vanno

considerati i riferimenti di valore di ognuna. Per ciascuno dei primi tre si batterà:

RR

poichè le funzioni @MAX, @MIN e @AVERAGE hanno tutti campi variabili (cioè relativi). L'ultima formula, invece, richiederà:

RN

La N (nessun cambiamento) indica che il riferimento a valore N13, le spese totali del periodo, è una costante della formula.

Converrà forse aggiungere un titolo e qualche sottolineatura per migliorare la chiarezza e l'aspetto del foglio. (Si usi il comando /IR per inserire righe nella parte superiore del foglio.) Le colonne che abbiamo creato in questo capitolo compaiono nella Figura 3.3; la si confronti con i dati del foglio elettronico per assicurarsi che nelle formule non ci sia qualcosa di sbagliato.

	A	N	O	P	Q	R
1		SPESE DOMESTICHE SETTIMANALI				
2		DAL 30 MAGGIO AL 21 AGOSTO				
3	==========	======	====	====	====	=======
4		TOTALI	MAX.	MIN.	MEDIE	%
5		======	====	====	====	=======
6	MISC	58.68	11.18	0.76	4.89	6.8
7	CEREALI	35.81	7.62	0.00	2.98	4.2
8	LATTERIA	49.53	11.18	0.00	4.13	5.8
9	GATTO	39.37	6.86	1.02	3.28	4.6
10	VARIE	91.46	11.18	4.83	7.62	10.7
11	VEGET.	143.76	17.02	5.84	11.98	16.8
12	SIGAR.	84.90	8.89	3.89	7.08	9.9
13	CARNE	152.27	22.86	4.06	12.69	17.8
14	DOLCI	99.36	12.09	6.35	8.28	11.6
15	LIQUORI	102.37	10.16	6.86	8.53	11.9
16	** TOTALI	857.51	76.71	64.26	71.46	100

Figura 3.3 - Le colonne con formule del foglio della spesa (vedi Figura 1.2)

LE FUNZIONI MATEMATICHE AVANZATE

Le funzioni matematiche analizzate in questa sezione rendono il VisiCalc utile per applicazioni scientifiche, tecniche e statistiche. Anche se si pensa di non utilizzare mai queste funzioni, è bene non saltare questa presentazione perchè vi vengono trattati punti di interesse generale.

Le funzioni trigonometriche ed esponenziali sono brevemente descritte nella Figura 3.4. Per illustrare un paio di queste funzioni in opera, in una formula piuttosto complessa, costruiremo un foglio elettronico della curva della "distribuzione normale". Probabilmente si sarà già avuto a che fare con la cosiddetta "curva a campana" che descrive la distribuzione statistica normale. Scriveremo una formula del VisiCalc per l'equazione di questa curva, poi useremo il comando /F* per crearne una rappresentazione grafica.

Funzione	Argomento	Descrizione
@EXP	valore	Eleva e alla potenza dell'argomento. (Per vedere il valore che il VisiCalc attribuisce ad e, battere @EXP(1).)
@LN	valore	Trova l'esponente naturale (base e)
@LOG10	valore	Trova il logaritmo in base 10.
@SQRT	valore	Trova la radice quadrata del valore.
@SIN, @COS, @TAN	valore (in radianti)	Le funzioni trigonometriche del seno, coseno e tangente. (Per trasformare i gradi in radianti usare: gradi *(PI/180)
@ATAN, @ACOS, @ASIN	valore	L'arcotangente, l'arcocoseno e l'arcoseno trigonometrici. (Tutte e tre le funzioni forniscono valori in radianti.)
@PI	nessun argomento	Dà il valore di pi greco.

Figura 3.4 - Le funzioni matematiche avanzate

La formula della distribuzione normale ha una potenza naturale al numeratore e la radice quadrata di pi greco al denominatore (1). Non ha importanza se il suo significato esatto non è chiaro; il punto essenziale di questo esempio è di dimostrare come sia possibile calcolare un'equazione complicata, e tracciarne un grafico.

---

(1)

$$y = \frac{e^{\frac{-x^2}{2}}}{\sqrt{2\pi}}$$

tramite il VisiCalc. La formula che utilizzeremo sul foglio elettronico è:

@EXP(—(A1^2)/2)/@SQRT(2*@PI)

Si noti come questa formula contenga tre funzioni: la potenza naturale, @EXP; la radice quadrata, @SQRT e la funzione che fornisce semplicemente il valore di pi greco, @PI. La funzione @PI è una di quelle che non richiedono argomento.

Prima di creare il foglio elettronico per la distribuzione normale, assicurarsi di aver memorizzato le informazioni attualmente presenti sul foglio che non si vogliono perdere (/SS), dopo di che battere /CY per cancellare lo schermo. Esamineremo la formula della distribuzione su un campo da —2 a +2, in incrementi di .2. Battiamo —2 nella posizione A1. In A2 inseriamo la formula:

+A1+.2

Replichiamo poi questa formula nelle posizioni da A3 a A21:

/R	{richiama il comando di replica}
(return)	{indica il campo di origine: A2...A2}
A3.A21 (return)	{indica il campo di destinazione: A3...A21}
R	{il riferimento a valore è relativo}

Portiamo adesso il cursore in B1 e scriviamo la formula della distribuzione normale:

@EXP(—(A1^2)/2)/@SQRT(2*@PI)

Controllare di aver inserito tutte le parentesi al posto giusto; questo è un esempio di formula che sarebbe molto diversa se le parentesi fossero altrove o mancassero. (E' chiaro il perchè?) Si replichi poi la formula sulla colonna B, da B2 a B21:

/R	{richiama il comando di replica}
(return)	{indica il campo di origine: B1...B1}
B2.B21 (return)	{indica il campo di destinazione: B2...B21}
R	{il riferimento a valore è relativo}

Il computer può impiegare un po' di tempo per effettuare questa replica, dato che si tratta di una formula complicata.

Adesso dividiamo verticalmente la finestra alla colonna C e aumentiamo la larghezza delle colonne della finestra di destra a 20 spazi:

```
>C1        {sposta il cursore a C1}
/WV        {divide verticalmente la finestra}
;          {porta il cursore nella finestra di destra}
/GC20      {porta la larghezza delle colonne a 20}
```

Ora siamo pronti per costruire il grafico. Forse si ricorderà com'è stato facile generare il grafico del Capitolo 2: abbiamo semplicemente riformattato una colonna di numeri (/GF*) nella finestra di destra. Questo è stato possibile perchè i numeri della colonna rientravano in un campo "comodo", da 1 a 12. Guardando la colonna B di questo foglio, si capisce che la situazione attuale è più complessa, poichè i numeri vanno da .05 a .4. Se ci limitassimmo ad attribuir loro un nuovo formato con il comando /GF* non otterremmo nessun grafico perchè i numeri sono troppo piccoli. Per questo è necessario aumentare proporzionalmente tutti i numeri moltiplicandoli per uno stesso fattore; continueranno a rappresentare la solita equazione e il VisiCalc sarà in grado di mostrarli in forma grafica.

Il fattore scalare viene scelto in base a quanti asterischi devono rappresentare il numero *più grande* del campo. Scegliamo arbitrariamente 15 e cioè vogliamo che il numero più grande della colonna B venga rappresentato da un rigo di 15 asterischi. Gli altri numeri verranno rappresentati da 15 asterischi o meno.

Per trovare il numero più grande del campo da B1 a B21 useremo la funzione @MAX, così che se moltiplichiamo ogni numero per

```
15/@MAX(B1...B21)
```

avremo un nuovo insieme di numeri che va da 0 a 15. La formula che inseriremo in C1 è quindi:

```
(15/@MAX(B1...B21))*B1
```

C'è però un'altra complicazione. Il comando /F* spezza i numeri, non li arrotonda, quando trasforma la parte intera in una serie di asterischi. Questo renderebbe la curva meno accurata, quindi dobbiamo usare la funzione @INT per arrotondare la formula:

```
INT((15/@MAX(B1...B21))*B1+.5)
```

che è finalmente l'equazione da scrivere nella posizione C1.

A questo punto, battiamo /F* per esprimere il risultato in formato grafico, dopo di che replichiamo la formula per creare la curva a campana:

```
/R         {richiama il comando di replica}
(return)   {indica il campo di origine: C1...C1}
C2.C21     {indica il campo di destinazione: C2...C21}
NNR        {il campo di @MAX è una costante; il riferi-
           mento a valore B1 è relativo}
```

Il risultato di questa replica dovrebbe essere quello della Figura 3.5.

	A	B		C
1	- 2	.0539910	1	**
2	-1.8	.0789502	2	***
3	-1.6	.1109208	3	****
4	-1.4	.1497275	4	******
5	-1.2	.1941861	5	*******
6	-1	.2419707	6	*********
7	-.8	.2896916	7	***********
8	-.6	.3332246	8	*************
9	-.4	.3682701	9	**************
10	-.2	.3910427	10	***************
11	0	.3989423	11	***************
12	.2	.3910427	12	***************
13	.4	.3682701	13	**************
14	.6	.3332246	14	*************
15	.8	.2896916	15	***********
16	1	.2419707	16	*********
17	1.2	.1941861	17	*******
18	1.4	.1497275	18	******
19	1.6	.1109208	19	****
20	1.8	.0789502	20	***
21	2	.0539910	21	**

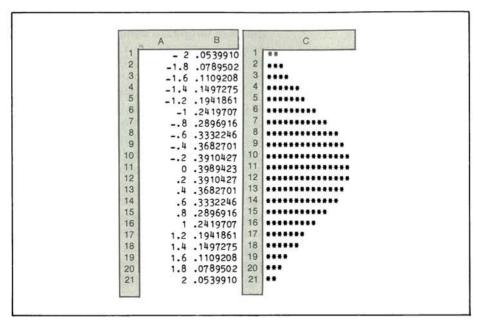



Figura 3.5 - La curva a forma di campana

SOMMARIO

Il VisiCalc è arricchito da un insieme di funzioni che possono essere incorporate nelle formule per svolgere molte applicazioni aritmetiche in modo più efficiente. La maggior parte delle funzioni richiede argomenti costituiti da un campo, da un elenco o da un valore. Questi argomenti possono essere espressi da numeri, da riferimenti a valori o da altre funzioni.

Le funzioni che vengono usate più spesso sono quelle semplici, come @SUM e @AVERAGE, però il VisiCalc ne ha anche altre, per utilizzazioni più specialistiche.

Nel Capitolo 4 esamineremo altre due categorie di funzioni, quelle di *ricerca*, @LOOKUP (guarda) e @CHOOSE (scegli), e le funzioni *logiche*.

CAPITOLO 4

LE FUNZIONI DEL VISICALC (PARTE II)

LE FUNZIONI @LOOKUP (RICERCA) E @CHOOSE (SCELTA)

In questo capitolo ci allontaneremo dalle vicissitudini del bilancio della spesa per considerare di nuovo i dieci venditori. Si ricorderà che aspettavano di sapere chi di

	A	B	C	D	E	F
1						
2						
3						
4		GEN	FEB	MAR	APR	MAG
5		===	===	===	===	===
6	BAKER	11.18	6.35	8.13	3.56	2.54
7	SMITH	7.62	7.11	5.59	3.30	0.25
8	FLINT	11.18	7.62	6.35	4.06	1.02
9	BROWN	1.52	1.78	3.05	4.83	6.86
10	VERN	4.83	5.33	7.62	9.91	9.91
11	MARLOW	11.43	11.68	13.97	10.67	10.67
12	HARPER	6.60	5.59	7.87	8.89	8.89
13	FLEMING	5.59	5.08	5.33	9.30	15.49
14	NASH	6.60	6.35	8.38	7.37	9.55
15	WHITE	6.86	7.37	9.40	8.38	8.89

	G	H	I	J	K	L	M
1							
2							
3							
4	GIU	LUG	AGO	SET	OTT	NOV	DIC
5	===	===	===	===	===	===	===
6	0.76	1.52	2.03	1.78	3.81	7.62	9.40
7	0.00	0.00	0.00	0.51	0.76	5.08	5.59
8	0.25	0.00	0.00	0.51	2.54	5.84	10.16
9	4.83	4.57	3.30	2.79	2.79	2.03	1.02
10	11.18	9.40	7.37	7.37	7.11	6.35	5.08
11	11.94	17.02	13.46	14.22	5.84	9.91	12.95
12	8.38	3.89	7.87	7.11	6.60	7.11	6.10
13	22.86	17.53	17.24	22.10	20.83	6.86	4.06
14	8.89	10.41	12.09	7.87	6.86	7.37	7.62
15	7.62	9.40	10.16	8.38	7.37	9.65	8.89

Figura 4.1 - Il foglio delle vendite con dodici mesi di dati

loro, a causa di risultati annuali poco convincenti, avrebbe dovuto mettersi a cercare un nuovo lavoro.

Costruiremo un foglio elettronico con i dati delle vendite di tutti i mesi; non utilizzeremo cioè il foglio del Capitolo 2 (che descriveva un solo mese) ma caricheremo i soli dati, memorizzati nel file che abbiamo chiamato NUMERI.VC, nel nuovo foglio. Alla sua sinistra inseriamo una colonna per i nomi dei venditori (Baker, Smith, Flint, Brown, Vern, Marlow, Harper, Fleming, Nash, White) e inseriamo cinque righe in alto per avere lo spazio per le intestazioni e i titoli. Nelle posizioni da B4 a M4 inseriamo abbreviazioni di tre lettere dei nomi dei mesi, in un formato giustificato a destra. Nella posizione A4 battiamo /TB, per fissare i titoli in entrambe le direzioni. La Figura 4.1 mostra il foglio elettronico a questo punto. Innanzi tutto calcoleremo le retribuzioni annue totali, compreso il premio di produzione, di ogni venditore. A tale scopo utilizzeremo un'altra funzione del VisiCalc, @LOOKUP (ricerca), che, a differenza di quelle esaminate finora, richiede due tipi di argomenti: un valore e un campo. Il suo formato è il seguente:

```
@LOOKUP(valore, campo)
```

Quasi certamente questa funzione è stata ispirata dall'insidiosa tabella che dobbiamo tutti studiare almeno una volta all'anno: quella delle aliquote fiscali. Il suo formato è ben noto: c'è una colonna di redditi a sinistra e una di imposte sulla destra. Si scende lungo la colonna dei redditi finchè non si trovano i due valori che individuano quello desiderato, poi si sposta il dito sulla destra e si guarda quanto si deve pagare di imposte.

@LOOKUP svolge proprio questa operazione di ricerca, solo che, naturalmente, la sua azione non è limitata alle tabelle delle tasse. Per utilizzarla si deve scrivere due colonne (o righe) di dati in qualche punto del foglio. Ecco che a questo punto torna la tabella dei premi di produzione che abbiamo già incontrato nel Capitolo 1.

Supponiamo di aver inserito la tabella dei premi nelle colonne M e N del foglio elettronico, subito sotto le righe con i dati sulle vendite:

	M	N
21	PREMI PROD. =	
22	- - - - - - -	- - - - -
23	0.00	0.25
24	50.00	1.00
25	75.00	1.50
26	100.00	2.25
27	125.00	3.00
28	150.00	3.50

La tabella funziona in questo modo: se uno dei venditori ha venduto supponiamo per $ 85.000, scendiamo lungo la colonna M finchè $ 85.000 (o 85, dato che i numeri sono espressi in unità di $ 1000) cade fra due valori. Trovato quel punto, ci spostiamo a destra, alla colonna N, sulla riga corrispondente al *minore* dei due delimitatori sulla colonna M, e troviamo il premio di produzione. Per vendite annue di $85.000, il premio di produzione è di $1,500 (dato che 85 è fra 75 e 100).

Per trovare questo valore tramite la funzione @LOOKUP va scritto:

```
@LOOKUP(85, M23...M28)
```

Il primo argomento, 85, è il valore che è oggetto della ricerca. Il secondo, M23...M28, è il campo di valori nel quale viene effettuata la ricerca. Se il campo è una colonna, come in questo esempio, @LOOKUP dà il valore corrispondente nella colonna immediatamente *a destra*, come mostra la tabella dei premi di produzione. (Se il campo è una riga, il valore viene dalla riga immediatamente *sottostante*.)

Prima di poter utilizzare la tabella dei premi di produzione dobbiamo avere una colonna delle vendite totali per ogni venditore; sono questi i valori in riferimento ai quali effettueremo la ricerca nella tabella. Inseriremo queste informazioni nella colonna N:

```
>N4/FR TOTALI          {intestazione nella posizione N4}
>N5/FR"= = = = = =     {sottolineatura dell'intestazione}
>(N6@SUM(B6.M6)        {formula dell'addizione in N6}
/R                     {richiama il comando di replica}
(return)               {campo di origine: N6...N6}
N7.N15 (return)        {campo di destinazione: N7...N15}
RR                     {B6 e M6 sono entrambi relativi}
```

A questo punto, se già non lo si è fatto, si dovrebbe inserire la tabella dei premi di produzione, a partire da M21. Siamo pronti a usare la funzione @LOOKUP per mostrare i premi nella colonna O. Cominciamo con un'intestazione:

```
>O4/FR PREMIO          {intestazione in O4}
>O5/FR"= = = = =       {sottolineatura intestazione}
```

La formula di ricerca, nella posizione O6, sarà la seguente:

```
@LOOKUP(N6,M23...M28)
```

Il VisiCalc prende il valore delle vendite totali in N6, ne trova la posizione nel campo di valori da M23 a M28 della tabella di ricerca e fornisce il corrispondente premio di produzione dai valori da N23 a N28. (In altre parole, il VisiCalc effettua

una ricerca fra i valori della colonna M e dà un valore preso dalla colonna subito a destra, la N.) Quando si scrive questa formula, in O6 dovrebbe apparire il valore 1.00. Per vendite totali di $ 58.680, il venditore Baker ottiene un premio di $ 1000.

Adesso possiamo replicare la formula di ricerca per trovare i premi per tutti i venditori. L'unica parte un po' complicata del procedimento di replica è la terza, quando va specificato se un riferimento a valore è relativo o costante. N6, la cifra delle vendite totali, è chiaramente relativa: il premio di ciascun venditore va determinato in base a una somma diversa. Invece i riferimenti a valore del campo M23...M28 sono costanti; usiamo la stessa tabella di riferimento per tutti. Ecco le istruzioni di replica (assicurarsi che il cursore sia in O6):

/R	{richiama il comando di replica}
(return)	{campo di origine: O6...O6}
O7.O15 (return)	{campo di destinazione: O7...O15}
RNN	{N6 è relativo; M23 e M28 non variano}

Per completare il foglio dobbiamo aggiungere una colonna con le retribuzioni totali. La retribuzione annua di base è $12.000 e la commissione è il 5%. Memorizziamo queste informazioni subito sopra la tabella dei premi, rispettivamente nelle posizioni N18 e N19:

	M	N
17	==========	==========
18	BASE	12.00
19	COMM.	0.05
20	----------	----------

La formula della retribuzione totale è:

premio + base + (commissione X vendite totali)

Inseriremo nel VisiCalc l'equivalente di questa formula nella posizione P6:

+O6+N18+(N19*N6)

che poi replicheremo per tutti i venditori:

/R	{richiama il comando di replica}
(return)	{campo di origine: P6...P6}
P7.P15 (return)	{campo di destinazione: P7...P15}
RNNR	{O6 e N6 sono relativi; N18 e N19 non variano}

Infine, si può aggiungere un titolo nella parte superiore del foglio elettronico nelle righe che abbiamo lasciato a tale scopo all'inizio. Le colonne della retribuzio-

ne compaiono nella Figura 4.2. Questo foglio verrà ampliato, quindi va memorizzato su disco prima di cancellare lo schermo.

Il VisiCalc ha un'altra funzione utilizzabile a fini di ricerca: @CHOOSE (scegli). Come @LOOKUP, anche @CHOOSE richiede due argomenti, nel seguente formato:

@CHOOSE(valore, elenco)

Si noti che @CHOOSE effettua una ricerca su un *elenco*, non su un *campo*. Un elenco, come si ricorderà, non dev'essere su una colonna o una riga contigua, ma può comprendere dati provenienti da varie fonti. @CHOOSE utilizza il valore che costituisce il primo argomento per determinare una posizione nell'elenco che è il secondo argomento e fornisce il valore indicato nell'elenco:

@CHOOSE(5,N7,P13,S1...S7)

@CHOOSE darà il quinto elemento dell'elenco costituito da N7, P13, S1...S7. (Quale sarà questo quinto elemento?)

	A	M	N	O	P
1		RETRIBUZ. ANNUE VENDITE			
2		DA GEN A DIC			
3	==========	==========	==========	==========	==========
4		DIC	TOTALI	PREMIO	TOT VEND
5		===	======	=====	=======
6	BAKER	9.40	58.68	1.00	15.93
7	SMITH	5.59	35.81	0.25	14.04
8	FLINT	10.16	49.53	0.25	14.73
9	BROWN	1.02	39.37	0.25	14.22
10	VERN	5.08	91.46	1.50	18.07
11	MARLOW	12.95	143.76	3.00	22.19
12	HARPER	6.10	84.90	1.50	17.75
13	FLEMING	4.06	152.27	3.50	23.11
14	NASH	7.62	99.36	1.50	18.47
15	WHITE	8.89	102.37	2.25	19.37
16					
17		========	========		
18		BASE	12.00		
19		COMM.	0.05		
20		========	========		
21		TABELLA PREMI			
22		--------	--------		
23		0.00	0.25		
24		50.00	1.00		
25		75.00	1.50		
26		100.00	2.25		
27		125.00	3.00		
28		150.00	3.50		

Figura 4.2 - Retribuzioni annue dei venditori

Le funzioni @LOOKUP e @CHOOSE sono chiaramente indispensabili nelle applicazioni in cui è necessario trovare un solo valore e ottenerne uno da una tabella o un elenco. Nel prosieguo di questo capitolo vedremo altre applicazioni della funzione @LOOKUP.

LE FUNZIONI LOGICHE

Le funzioni logiche del VisiCalc costituiscono una categoria a parte. Al primo impatto possono intimidire un po', ma in realtà usarle è molto facile:dette funzioni aumentano significativamente le possibilità di controllo sui numeri che si inseriscono nel foglio elettronico.

L'oggetto principale delle funzioni logiche non sono valori numerici, ma *valori logici*, che sono solo due: *vero* o *falso*. Nel VisiCalc, le espressioni logiche vengono utilizzate per effettuare confronti fra due valori. I "verbi" delle espressioni logiche, cioè i simboli che indicano il confronto che viene effettuato, sono i seguenti:

=	{è uguale}
<	{è minore di}
>	(è maggiore di)
<=	(è minore di o uguale a)
>=	(è maggiore di o uguale a)
< >	(è diverso da)

Questi simboli vengono talvolta chiamati *operatori logici*. Ecco alcuni esempi di espressioni che utilizzano questi operatori:

```
B1 = 5
C7 < = M3
A1 < > A2
G11 > 10
```

Ciascuna espressione è vera o falsa. Espressioni come queste possono essere inserite direttamente nelle funzioni logiche del VisiCalc e danno valori logici di vero o falso.

Le principali funzioni logiche sono riassunte nella Figura 4.3; probabilmente la più importante è @IF (se), quella che viene utilizzata più spesso. La sua struttura è simile a quella delle istruzioni IF di alcuni linguaggi di programmazione, come il BASIC o il Pascal. In questi linguaggi l'istruzione fornisce due alternative di comportamento la cui scelta dipende dalla verità o dalla falsità di un'espressione logica:

```
IF (espressione logica) THEN (azione #1) ELSE (azione #2)
```

Funzione	**Argomento**	**Descrizione**
@IF	(valore logico),valore,valore	Se il valore logico è vero, la funzione dà il primo valore dell'argomento; se è falso, il secondo.
@AND	elenco di valori logici	Se *tutti* i valori logici dell'argomento sono veri, la funzione dà il valore vero altrimenti dà falso
@OR	elenco di valori logici	Se *uno o più* dei valori logici dell'argomento sono veri, la funzione dà vero se *tutti* i valori logici sono falsi, la funzione dà falso.
@NOT	valore logico	Dà falso se il valore logico dell'argomento è vero dà vero se il valore logico è falso.
@TRUE	nessun argomento	Dà il valore vero; può essere utilizzata come argomento di una funzione logica o per mostrare TRUE (vero) sul foglio.
@FALSE	nessun argomento	Dà il valore falso; può essere utilizzata come argomento di una funzione logica o per mostrare FALSE sul foglio.

Figura 4.3 - Le funzioni logiche

Se l'espressione logica è vera viene effettuata l'azione #1; se è falsa, la #2.
La funzione IF del VisiCalc ha il seguente formato:

```
@IF(valore logico,valore,valore)
```

Inserisce uno dei due valori in una posizione del foglio a seconda del valore logico. Per esempio, supponiamo di aver inserito in B2 la seguente formula:

```
IF(B1>1000,25,10)
```

Questa espressione può essere letta come segue: "Se il valore nella posizione B1 è maggiore di 1000, inserire 25 in B2; altrimenti (se B1 è inferiore o uguale a 1000)

mettere 10 in B2". In altri termini, se il valore corrente di B1 è al di sopra di 1000, per esempio 1500, sul foglio elettronico ci sarà:

	B
1	1500
2	25

Se invece B1 è inferiore o uguale a 1000, per esempio 750, si avrà:

Talvolta sarà utile far dipendere l'azione di una funzione @IF da espressioni logiche più complicate. Si potrà voler combinare due (o più) uguaglianze o disuguaglianze per decidere che valore ottenere dalla funzione @IF. In questi casi si può utilizzare le funzioni @AND (e) e OR (o) come parte dell'argomento della funzione @IF.

Facciamo un rapido esempio. Supponiamo di dover decidere dove andare quest'anno per le vacanze estive e di aver fatto un elenco di diversi elementi (clima, costo, impianti turistici ecc.) su cui si vuol basare la decisione. Si attribuiscono punti a ognuna della mezza dozzina di destinazioni potenziali a seconda di come rispondono agli elementi di valutazione. Si inserisce questo sistema di votazioni su un foglio del VisiCalc e quando tutti i punteggi sono stati stabiliti, li si esaminerà, forse aggiungendo qua e là un fattore preferenziale (che possiamo fornire noi, non il computer) e poi si compie la scelta.

Per il criterio del clima si è introdotto quello che si considera il campo di temperatura ideale, da 25 a 33, nelle posizioni A19 e A20:

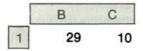
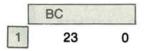
	A
18	TEMP.
19	25
20	33

Si assegnano 10 punti a tutte le località che rientrano in questa categoria. Nella colonna B del foglio si inserisce la temperatura media di ogni città e i punti ottenuti vengono registrati nella colonna C.

Nella posizione C1 si inserisce la seguente formula:

```
@IF(@AND(B1>=A19,B1<=A20),10,0)
```

La funzione @AND è un'espressione logica che fornisce il valore vero o quello falso. @AND dà vero se *tutte* le espressioni del suo argomento sono vere. Se *una qualsiasi* di quelle espressioni (anche una soltanto!) è falsa, @AND dà il valore falso. Possiamo quindi leggere questa formula del VisiCalc come segue: "Se il valore in B1 è maggiore o uguale al valore in A19 *e* minore o uguale al valore in A20, nella posizione C1 va inserito il valore 1, altrimenti va messo il valore 0". In altri termini, se la temperatura rilevata in B1 rientra nel campo ideale delle temperature che appare in A19 e A20, la località di villeggiatura ottiene 10 punti, in caso contrario non ne ottiene nessuno.

Possiamo controllare la correttezza di questa formula inserendo in B1 valori diversi. Cominciamo con 29, una temperatura nel campo desiderato. Ecco cosa vedremo:

I punti sono stati assegnati nel modo corretto. Ora proviamo con 23:

Fa troppo freddo; niente punti. Infine proviamo con 35:

	BC	
1	35	0

Troppo caldo; niente punti. Quindi la funzione @IF in combinazione con @AND funziona come deve.

La funzione OR (o) dà il valore vero se *almeno una* delle espressioni logiche del suo argomento è vera. Risulta che avremmo potuto ottenere esattamente gli stessi risultati nell'attribuzione dei punti in relazione alla temperatura se avessimo utilizzato la funzione OR al posto di @AND:

@IF(@OR(B1<A19,B1>A20),0,10)

In questo caso OR dà vero se la temperatura in B1 è *al di fuori* del campo desiderato. Si noti che i due argomenti di valore della funzione @IF sono stati invertiti: vero significa niente punti; falso, 10.

Torneremo adesso, per l'ultima volta, al foglio dei venditori per un secondo esempio della funzione @IF. Assumeremo che la situazione sia questa: è necessario stabilire un criterio oggettivo per individuare i venditori che non hanno un rendimento soddisfacente. Naturalmente questo criterio verrà probabilmente integrato da giudizi più soggettivi prima che venga decisa l'azione da intraprendere, però almeno il criterio oggettivo può essere sviluppato sul foglio elettronico del VisiCalc.

Tale criterio sarà basato su tre medie che verranno calcolate dai dati del foglio elettronico delle vendite:

1. la media delle vendite annue totali dei venditori,

2. la media delle migliori vendite mensili dei venditori (cioè, per ogni venditore si trovano le vendite effettuate nel suo mese migliore e poi si calcola la media per tutti i venditori)

3. la media delle peggiori vendite mensili dei venditori.

Qualsiasi venditore le cui prestazioni non siano pari o superiori alle medie in almeno una delle tre categorie verrà posto in osservazione.

La prima delle tre medie può essere calcolata direttamente dalla colonna delle vendite annue totali, la N, che abbiamo incluso nel foglio elettronico in questo capitolo. Metteremo questa cifra nella posizione R18. Ecco la formula:

```
@AVERAGE(N6...N15)
```

Può essere utile identificare questo valore con un'etichetta nella posizione Q18:

	Q	R
18	MED.TOT =	85.75

Per calcolare le altre due medie dovremo stabilire colonne per i migliori e i peggiori risultati mensili per ogni venditore. Per costruirle, utilizzeremo le funzioni @MAX e @MIN. I valori delle migliori prestazioni mensili andranno nella colonna Q, quelli delle peggiori nella colonna R. Ecco la sequenza di istruzioni per le due colonne:

>Q4/FR MIGL	{colonna di intestazione giustificata a destra}
>Q5/FR"= = = =	{sottolinea intestazione}
>@Q6MAX(B6.M6)	{formula del valore massimo}
>R4/FR PEGG	{colonna di intestazione giustificata a destra}
>R5/FR"= = = =	{sottolinea intestazione}
>R6@MIN(B6.M6)	{formula del valore minimo}
>Q6/R	{richiama comando di replica}
.R6 (return)	{campo di origine: Q6...R6}
Q7.Q15 (return)	{campo di destinazione: Q7...Q15}
RR RR	{B6 e M6 sono relativi in entrambe le formule}

Si noti che le formule sono state replicate tutte e due insieme.

Metteremo poi le medie delle colonne "migliori" e "peggiori" nelle posizioni Q17 e R17. Le formule sono

```
>Q17 @AVERAGE(Q6...Q15)
>R17 @AVERAGE(R7...R15)
```

Per identificare questi valori può darsi che convenga andare alla posizione A17 e inserire un'etichetta tipo "**MEDIA". Le medie dovrebbero avere questo aspetto:

	A	Q	R
4		MIGL.	PEGG.
5		===	===
.			
.			
.			
16			
17	**MEDIA	11.90	3.36
18		MED.TOT =	85.75

Si sono così calcolate le tre medie su cui sarà la basata la formula di valutazione dei venditori. Portiamo il cursore in S4 e battiamo l'intestazione della colonna "<MEDIA" ("minore della media") e sottolineiamola nella posizione S6. Qual è il modo migliore per indicare quali venditori hanno prestazioni al di sotto della media? Uno buono può consistere nel mettere semplicemente un indicatore di qualche genere nelle righe che contengono tali valori. Abbiamo visto che /F* è un comodo comando di formattazione che cambia i numeri in asterischi; forse lo si può utilizzare per generare gli indicatori.

Portare il cursore nella posizione S6 e battere /F*. La formula @IF può essere scritta associata a @AND o associata a @OR, ma in questo caso sembra più appropriato @AND. Scriveremo la formula per il primo venditore, Baker, e la replicheremo sulla colonna per gli altri.

Affinchè un venditore venga posto in osservazione, devono verificarsi tre condizioni: le vendite annue totali di quella persona devono essere inferiori alle vendite annue medie per venditore:

```
N6<R18
```

le vendite del mese migliore di quel venditore devono essere inferiori alla media

delle vendite migliori:

Q6<Q17

e le vendite del mese peggiore del venditore devono essere inferiori alla media delle vendite peggiori:

R6<R17

Riuniremo le tre condizioni in una funzione @AND:

@AND(N6<R18,Q6<Q17,R6<R17)

Se il valore di questa funzione @AND è vero, le prestazioni di vendita della persona considerata sono inferiori alla media in tutte e tre le categorie e questo fatto lo vogliamo segnalare. Ecco come genereremo l'indicatore: scriveremo la funzione @IF in modo che nella posizione venga posto un 2 se il valore è vero e 0 se il valore è falso. Nel formato /F*, il due diventerà un doppio asterisco, mentre lo 0 rimarrà vuoto. La formula da scrivere in S6 è quindi:

@IF(@AND(N6<R18,Q6<Q17,R6<R17),2,0)

Questa funzione @IF contiene i tre argomenti richiesti: il valore logico proviene dalla funzione @AND; il valore corrispondente a *vero* è 2, quello corrispondente a

	A	Q	R	S
1	VALUTAZIONI DEI VENDITORI			
2				
3	===============================			
4		MIGL.	PEGG.	< MED.
5		====	=====	======
6	BAKER	11.18	0.76	**
7	SMITH	7.62	0.00	**
8	FLINT	11.18	0.00	**
9	BROWN	6.86	1.02	**
10	VERN	11.18	4.83	
11	MARLOW	17.02	5.84	
12	HARPER	8.89	3.89	
13	FLEMING	22.86	4.06	
14	NASH	12.09	6.35	
15	WHITE	10.16	6.86	
16				
17	**MED.	11.90	3.36	
18		MED.TOT=	85.75	

Figura 4.4 - La valutazione dei venditori

falso è 0. Nella posizione S6 del foglio elettronico dovrebbe comparire un asterisco doppio, a indicare che la prestazione di Baker è inferiore alla media.

Infine, replicheremo questa formula per riferirla a tutti i venditori. Il campo di destinazione va da S7 a S15. Nella terza fase della replica bisogna stare attenti ai riferimenti a valori; tre sono relativi (i valori delle vendite individuali) e tre rimangono costanti per tutto il procedimento di replica (le medie):

RNRNRN

I risultati sono nella Figura 4.4. Si noti il titolo che è stato aggiunto nella riga 1, sopra le tre colonne di informazione che abbiamo creato in questa sezione. Poichè un lungo foglio elettronico appare sullo schermo a sezioni, non è un problema includere più titoli diversi per descrivere le varie parti del foglio.

COME FARE SENZA LE FUNZIONI LOGICHE

Alcune versioni del VisiCalc non offrono le funzioni logiche; in questi casi si dovrà essere un po' più creativi per effettuare i confronti e le operazioni che esse svolgono.

Che si abbia o meno a disposizione una versione del VisiCalc con le funzioni logiche, questo programma va sempre visto nel più ampio contesto del personal computer su cui opera. Talvolta ci saranno applicazioni che, sebbene risolvibili tramite il VisiCalc, risultano enormemente più semplici se affrontate tramite un linguaggio di programmazione di alto livello, come il BASIC o il Pascal. E' vero che circolano virtuosi del VisiCalc che vorrebbero far credere che il programma sia la risposta definitiva a tutte le necessità di programmazione, però questa filosofia può tornare a svantaggio del VisiCalc. In molti esempi, abbiamo visto come la sua eleganza risieda nella semplicità, ma questo è vero solo delle applicazioni per le quali è stato progettato.

In generale, un linguaggio come il BASIC è molto più adatto a compiti che richiedono una logica complicata e più possibilità decisionali. In questi casi si dovrebbe considerare la possibilità di usare il VisiCalc per l'immissione dei dati, di creare un file DIF per la loro memorizzazione su disco e poi di scrivere un programma in BASIC per elaborarli. Costruire programmi multi-uso in BASIC per leggere e scrivere file DIF è piuttosto semplice (come vedremo nei Capitoli 6, 7 e 8). Quando si hanno a disposizione tali programmi, può darsi che si scopra che il VisiCalc e il BASIC sono partner ideali per molte applicazioni elaborative.

Ma che succede se si deve svolgere un lavoro che impone alcune semplici decisioni logiche che rientrano nel campo di applicazione del VisiCalc, e la versione del programma di cui si dispone non ha le funzioni logiche? Spesso, pensandoci un po', si arriva a soluzioni adeguate che utilizzano altre funzioni del VisiCalc. Guardiamo alcuni esempi.

Le funzioni @MAX e @MIN possono essere utilizzate al posto di certe disugua-

glianze @IF anzi, talvolta @MAX e @MIN sono molto più dirette. Per esempio, le due formule che seguono danno gli stessi risultati:

```
@IF((A1<100),A1,100)
@MIN(A1,100)
```

In entrambi i casi, se il valore in A1 è minore di 100 la funzione dà A1, altrimenti dà 100.

In situazioni un po' più complicate, @IF può essere simulata da @LOOKUP. La seguente espressione che utilizza @IF costituisce un esempio:

```
@IF(A1<100,10,25)
```

Questa espressione dà 10 se il valore in A1 è inferiore a 100 o 25 se A1 è maggiore o uguale a 100. Possiamo costruire una tavola di esame con i valori 10 e 25 nella colonna di destra:

	C	D
1	0	10
2	100	25

La formula che segue utilizza questa tavola per ottenere gli stessi risultati della precedente espressione con @IF:

```
@LOOKUP(A1,C1...C2)
```

Si può vedere cosa succederebbe per valori di A1 diversi: finchè A1 è inferiore a 100, @LOOKUP dà il valore della posizione D1, cioè 10. Per valori di A1 maggiori o uguali a 100, @LOOKUP dà il valore in D2, cioè 25. Si batta questa formula in un foglio per far pratica, nella posizione E1, e si facciano esperimenti con diversi valori di A1. Durante questo procedimento, può darsi che si scopra un problema: l'espressione con @LOOKUP funziona solo se A1 è maggiore o uguale a zero; se A1 contiene un valore minore di 0, @LOOKUP dà il messaggio NA, che significa *Not Available* (non disponibile) e non il valore in E1. Questo avviene perchè il campo della tavola di ricerca non comprende i numeri negativi. Per rimediare alla situazione basta modificare la tavola specificamente, C1 deve contenere un numero negativo sufficientemente piccolo da coprire ogni situazione prevista.

Continuiamo a fare esperimenti con questa formula sostituendo il valore attualmente in C1 con -9999999. Proviamo adesso a mettere qualche piccolo numero negativo nella posizione A1: adesso la formula con @LOOKUP simulerà correttamente la formula con @IF dando il valore 10.

Si può anche usare la funzione @LOOKUP, in combinazione con la funzione di valore assoluto, @ABS, in sostituzione delle *uguaglianze* nelle espressioni con @IF.

Supponiamo di voler verificare se due valori, nelle posizioni A1 e A2, sono uguali. Se lo sono, dobbiamo mettere un 1 in B1, se non lo sono, uno 0. L'espressione @IF che darebbe questo risultato è:

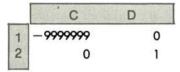
@IF(A1=A2,1,0)

Per ottenere gli stessi risultati senza @IF possiamo sottrarre un valore dall'altro: se la differenza è zero, sappiamo che sono uguali. Se però non sono uguali, il risultato della sottrazione può essere positivo o negativo, a seconda di quale dei due numeri è maggiore. Per poter usare quel risultato con una funzione @LOOKUP, sarebbe comodo che la differenza fosse, per esempio, sempre negativa, e per far questo si può usare la funzione di valore assoluto.

Si ricorderà che la funzione ABS dà il valore assoluto, quindi positivo, di un numero, perciò il risultato della seguente espressione sarà sempre negativo, tranne quando A1 è uguale a A2:

—@ABS(A1-A2)

Si può creare una tavola di ricerca che dia il valore 0 per i numeri negativi e il valore 1 per lo zero:

	C	D
1	−9999999	0
2	0	1

Come nel caso della tavola precedente, dobbiamo stare attenti che il valore negativo in C1 sia sufficientemente piccolo da tener conto di tutte le situazioni previste. Adesso si può scrivere la formula:

@LOOKUP(-@ABS(A1-A2),C1...C2)

Inseriamo questa formula nella posizione E1 e controlliamola con alcuni valori in A1 e A2. Poichè abbiamo la funzione @ABS, non importa quale dei due valori è maggiore; questa formula verificherà l'uguaglianza di due numeri qualsiasi, compresi quelli negativi e lo zero.

Per valutare le uguaglianze è stato suggerito anche un altro metodo, (1) che è forse più comodo (non richiede una tavola di consultazione), ma è meno affidabile

---

Nota
(1) SATN, Vol. 1, n. 4, marzo/aprile 1982, p.6.

di quello che utilizza @LOOKUP e @ABS. Ecco la formula:

@INT(A1/A2)*@INT(A2/A1)

Questa formula è basata sul fatto che il valore intero di una frazione fra 0 e 1 è 0. Così se A1 non è uguale ad A2 una delle due espressioni @INT darà 0; se invece A1 è uguale ad A2 la formula darà il valore 1. Questo metodo ha due limiti, uno evidente e uno meno chiaro:

1. Se A1 o A2 è uguale a zero, l'espresssione dà un errore: sul foglio comparirà il messaggio ERROR. (0 al denominatore di una frazione è illegale.)

2. Se A1 è uguale a -A2, la formula darà 1, indicando un'uguaglianza. Per esempio, se A1 contiene 7 e A2 -7, la formula darà -1.

Senz'altro si escogiteranno, o si leggeranno, molti altri modi per simulare le funzioni logiche. Alcuni avranno un'utilità generale, essendo utilizzabili per molte applicazioni diverse; altri permetteranno di risolvere un problema specifico. Con un po' di ingegnosità è possibile prendere la maggior parte delle decisioni semplici con o senza l'aiuto delle funzioni logiche del VisiCalc.

SOMMARIO

Le funzioni di ricerca, @LOOKUP e @CHOOSE, sono comode per trovare un valore di una tabella o di un elenco. Le funzioni logiche, invece, operano su espressioni logiche ed effettuano una scelta fra due valori. Ci sono molti metodi per ottenere gli stessi risultati nelle versioni del VisiCalc che non dispongono delle funzioni logiche.

Quando si usano queste funzioni, si deve stare attenti a utilizzare il VisiCalc per quello per cui è stato progettato. Può essere tentante cercare di svolgere con il VisiCalc un compito che sarebbe più semplice con altri strumenti di programmazione.

Nel Capitolo 5 esamineremo alcuni concetti generali sulla messa a punto dei fogli elettronici e introdurremo molte altre funzioni del VisiCalc.

CAPITOLO 5

COSTRUZIONE DEI FOGLI ELETTRONICI DEL VISICALC

FOGLI ELETTRONICI GENERALI: FORMULE SENZA DATI

Dopo quattro capitoli la maggior parte degli aspetti meccanici per utilizzare il programma VisiCalc è ormai stata esaminata. Prima di procedere, comunque, conviene riflettere sulla *struttura* dei fogli elettronici, che è l'argomento di questo capitolo.

Due sono gli approcci alla programmazione dei computer. (Sì, l'utilizzazione del VisiCalc *può* rientrare nella categoria della programmazione. Un programma può essere considerato un insieme di istruzioni impartite al computer per svolgere un compito. In questo senso, i comandi e le formule tramite cui si costruisce un foglio elettronico del VisiCalc costituiscono un programma.) Un approccio privilegia le necessità immediate e le soluzioni rapide. Secondo questa filosofia, un programma, o un foglio del VisiCalc, vengono messi a punto per una sola utilizzazione, per risolvere un problema particolare. Non si intende memorizzare il programma per riutilizzarlo, ma si vuole solo trovare, *rapidamente*, una risposta a una domanda.

Questo approccio, intenzionalmente miope, è spesso giustificato. Molto più spesso, tuttavia, quando il computer viene utilizzato, giorno dopo giorno, per le stesse operazioni, è evidentemente più utile un approccio di carattere più generale. Si tratta di un modo di procedere più impegnativo e, a breve termine, molto più lungo. A lungo termine, però, è molto più efficace e permette di risparmiare ore, giorni o settimane di programmazione.

La chiave di questo secondo approccio è di pensare sempre in termini di creazione di strumenti permanenti, piuttosto che di soluzione dei problemi immediati. La sua difficoltà è subito evidente: sotto la pressione di far fronte alle scadenze di lavoro quotidiane, è spesso difficile pensare al lungo termine, anche quando *si sa* che lavorando pochi minuti in più oggi si può risparmiare un'ora domani.

La maggior parte dei programmatori, compresi gli utenti dei personal computer, vengono convinti della bontà di questo approccio dall'esperienza: dopo aver scritto cinque volte lo stesso programma per risolvere situazioni solo poco diverse fra loro, ci si comincia a rendere conto che il problema va considerato da un punto di vista più generale. Scrivere un programma utilizzabile molte volte significa che, alle

lunghe, se ne possono dimenticare i dettagli, concentrandosi sui dati da fornire e sulle risposte ottenute. Significa anche che saremo *noi* a controllare il computer, non l'inverso.

E' chiaro che questa discussione sui due approcci alla programmazione si riferisce tanto al VisiCalc che a qualsiasi altro strumento di programmazione. Con il VisiCalc, il modo più evidente per rendere generico un foglio elettronico, cioè, per renderlo riutilizzabile, consiste nel costruire una struttura che sia indipendente dai dati che vi verranno poi immessi. In altri termini, si deve progettare i fogli elettronici in modo da poter facilmente modificare i dati di input essenziali e successivamente osservare i cambiamenti che ne derivano nei dati calcolati. Questo tipo di progettazione richiede un'attenta pianificazione iniziale, forse con carta e penna, prima di cominciare a inserire nel foglio elettronico le istruzioni. Le formule vanno costruite essenzialmente con riferimenti a valori, piuttosto che con dati numerici. Se è probabile che le formule rimangano immutate finchè si utilizza un determinato foglio, è invece molto probabile che i dati cambino; se tutti i valori sono espressi come riferimenti a valori, sarà molto facile modificare il foglio per immettervi nuovi dati.

Abbiamo rapidamente toccato questo punto nel Capitolo 2. Forse si ricorderà come il foglio dei venditori sia stato reso più generale tramite una piccola tabella distinta contenente lo stipendio di base e le commissioni. Quando questi dati sono stati modificati, è stato facile vedere cosa succedeva alla colonna delle retribuzioni.

Il programma VisiCalc offre diversi strumenti per semplificare la costruzione di fogli più generici. Questi comprendono quattro funzioni - @NA, @ERROR, @ISAN e @ISERROR - e due comandi globali, importanti e piuttosto sottili - /GO e /GR. In questo capitolo esamineremo tutti questi elementi. Inoltre, per illustrare i concetti della struttura generalizzata, costruiremo un piccolo foglio elettronico che utilizza la funzione @NPV. Parleremo del significato del valore attuale netto e della sua utilizzazione nei prospetti commerciali. La Figura 5.1 riassume le cinque nuove funzioni che apprenderemo in questo capitolo.

LA FUNZIONE @NA E IL COMANDO /GR

Spesso un foglio elettronico contiene dati che definiscono, o contribuiscono a definire, parte o tutto il contenuto del foglio. E' stato il caso del foglio dei venditori, in cui lo stipendio di base e la percentuale delle commissioni determinavano il contenuto della colonna delle retribuzioni. Tali valori essenziali vengono spesso chiamati *parametri*; se sono isolati in una tavola completa di etichette, questa viene chiamata *tavola dei parametri*, ed è lì che si immagazzinano o si modificano i dati che definiscono un foglio elettronico. I *risultati* di questi valori, appaiono invece in qualche altra parte.

Funzione	Argomento	Descrizione
@NA	nessuno	Mostra il messaggio NA (da *Not Available*, non disponibile) in una posizione. Se una formula fa riferimento a una posizione contenente NA, darà anch'essa NA.
@ISNA	valore	Dà vero se il valore del suo argomento è NA, altrimenti dà falso.
@ERROR	nessuno	Mostra il messaggio ERROR in una posizione se una formula fa riferimento a una posizione contenente ERROR, anche il suo risultato sarà ERROR
@ISERROR	valore	Dà il valore vero se il valore del suo argomento è ERROR, altrimenti dà falso.
@NPV	valore, campo	Usando il valore dell'argomento come *tasso di sconto*, trova il *valore attuale netto* dei valori del campo il tasso di sconto va espresso in forma decimale, non percentuale.

Figura 5.1 - Funzioni di vario tipo

Talvolta, quando si utilizza il foglio, alcuni, o tutti, i valori dei parametri non sono disponibili. Oppure si può voler lasciare vuota l'intera tavola, mentre si mettono a punto le formule. Per queste situazioni, il programma VisiCalc fornisce la funzione @NA, dove @NA sta per *Not Available* (non disponibile). E' una funzione che non richiede argomenti: se la si inserisce in una posizione del foglio, il VisiCalc vi mostra semplicemente il messaggio

NA

Ancor più pertinente, se una formula contiene un riferimento a un valore di una posizione che al momento contiene NA *anche* il risultato della formula sarà NA.

Facciamo un piccolo esperimento: mettiamo un valore nella posizione A1 e una percentuale in A2 e scriviamo in A3 una formula per moltiplicare il valore per la percentuale:

+A1*(A2/100)

Cominciamo inserendo un valore qualsiasi, per esempio 450, in A1. Spostiamo poi il cursore in A2, ma invece di battere un valore, scriviamo @NA. Si noti sul rigo

dei contenuti come questo venga accettato come un valore:

A2 (V) @NA

Spostiamo poi il cursore in A3 e inseriamo la formula precedente: il rigo dei contenuti mostra la formula in modo corretto, ma nella posizione c'è NA. Ecco cosa dovrebbe esserci sul foglio:

A3 (V) +A1*(A2/100)

	A
1	450
2	NA
3	NA

Il VisiCalc è pronto per calcolare il risultato della formula in A3 non appena viene fornito un valore per A2, ma fino a quel momento, poichè la formula dipende dai dati in A2, il risultato comparirà come NA.

Si inserisca il valore 10 in A2 e si guardi cosa succede. La formula in A3 fornisce immediatamente un valore:

	A
1	450
2	10
3	45

La funzione @NA può essere preziosa per mettere a punto un foglio generale. Quando si costruisce una tavola dei parametri, questi possono tutti inizialmente avere il valore @NA, dopo di che si può scrivere le formule come se la tavola contenesse già i valori desiderati. Il risultato di qualsiasi formula che faccia riferimento ai valori dei parametri della tavola sarà semplicemente NA.

Per illustrare la funzione @NA in modo più significativo, costruiremo un piccolo foglio per esaminare il rendimento futuro di un investimento. Questo foglio finirà per contenere tre sezioni di dati distinti: una tavola di parametri, un campo di valori annuali intermedi e una coppia di posizioni che riassumono il valore totale dell'investimento. La terza sezione costituisce il rigo riepilogativo, la risposta alla domanda: *quanto vale l'investimento*.

All'inizio la tavola dei parametri contiene tre valori: la data del primo anno del piano di investimento (per esempio, 1983), il rendimento previsto durante il primo anno e la prevista percentuale di aumento o di diminuzione del rendimento nei successivi anni del piano. Tutti e tre i valori saranno inizialmente NA:

PRIMO ANNO DEL PIANO	NA
RENDIMENTO DURANTE NA	NA
% ANNUA DI VARIAZIONE	NA

In questo foglio esamineremo il rendimento dell'investimento per cinque anni; nei paragrafi che seguono, la costruzione del foglio verrà esaminata passo per passo.

Per gettare le basi del foglio, impartiremo due comandi globali e tracceremo delle linee per separare le diverse sezioni. Uno dei vantaggi evidenti di questo modo di procedere è che le operazioni di "messa a punto" possono essere simultanee e quindi richiedono meno tempo. Per cominciare, quindi, battiamo la seguente serie di comandi:

```
/GC7                 {riduce la larghezza delle colonne a 7 spa-
                     zi}
/GF$                 {formato globale in dollari e centesimi}
>A1/-=               {traccia le linee nel foglio}
>A3/-=
>A10/-=
>A13/FR"----
>A16/-=
>A1                  {va in A1 per la replica delle linee}
/R                   {richiama il comando di replica}
.A16 (return)        {campo di origine: A1...A16}
B1.E1 (return)       {campo di destinazione: B1...E1}
```

La figura 5.2 mostra i risultati di questi comandi. Le linee dividono il foglio in quattro sezioni: il titolo del foglio, la tavola dei parametri, i valori annuali e i totali.

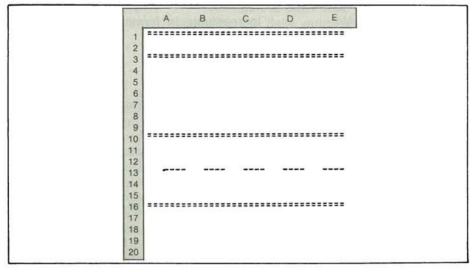



Figura 5.2 - L'intelaiatura del foglio degli investimenti.

Abbiamo usato il formato globale in dollari e centesimi per tutto il foglio anche se alcuni valori avranno un formato diverso; sarà necessario apportare modifiche singole.

Il passaggio successivo consiste nell'inserire il titolo, la tavola dei parametri e @NA per i valori dei parametri. Poichè la larghezza delle colonne è stata fissata in 7 spazi, ogni etichetta dovrà essere inserita a 7 caratteri per volta; alcune etichette non entreranno tutte in una posizione:

```
>B2 STATO I          {riga 2: il titolo}
>C2 NVESTIM
>D2 ENTI
>A5 PRIMO A          {riga 5: prima etichetta dei parametri}
>B5 NNO PIA
>C5 NO
>D5/FI @NA           {data in formato intero; ancora non disponi«
                     bile}
>A6 RENDIME          {riga 6: seconda etichetta dei parametri}
>B6 NTI DEL
>C6 /FI+D5           {l'anno viene dalla posizione D5}
>D6 @NA              {valore del parametro non ancora disponi«
                     bile}
>A7 % ANNUA          {riga 7: terza etichetta dei parametri}
>B7 VARIAZ.
>C7 E
>D7 /FG @NA          {percentuale in formato generale; valore
                     non disponibile}
```

La Figura 5.3 mostra il foglio in questo momento. Il termine *stato* del titolo fa riferimento al concetto del "cosa succede se" tipico del foglio elettronico: ogni investimento può essere studiato da punti di vista diversi modificando semplicemente i valori della tavola dei parametri. Si noti come tutti i valori siano NA. Si osservi in particolare la posizione C6, che fa parte dell'etichetta del secondo parametro: non appena il valore sarà disponibile nella posizione D5, mostrerà la data del primo anno.

Passiamo ora alla sezione dei dati intermedi. La data e il rendimento del primo anno vengono direttamente dalla tavola dei parametri:

```
>A12/FI +D5          {data del primo anno}
>A14+D6              {rendimento del primo anno}
```

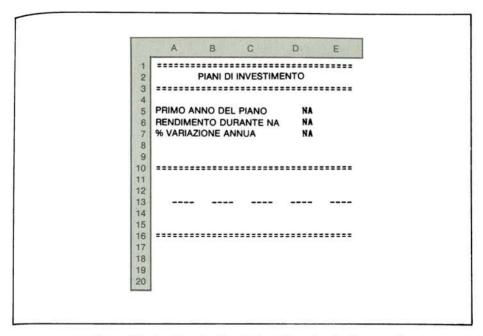



Figura 5.3 - Inserimento di un titolo e di una tavola di parametri

I valori successivi vanno calcolati e replicati:

>B12 /FI +A12+1	(incremento della data)
>B14 +A14*(1+(D7/100))	{formula delle variazioni annue del rendimento}
>B12	{va a B12}
/R	{richiama il comando di replica}
.B14 (return)	{campo di origine: B12...B14}
C12.E12 (return)	{campo di destinazione: C12...E12}
R RN	{A12 e A14 sono relativi; D7 non cambia}

Finora tutti i valori sono NA, come mostra la Figura 5.4. Inseriamo adesso alcuni valori nella tavola dei.parametri. Cominciamo con l'anno, in D5:

>D5 1983 (inserimento della data in D5)

Appena viene effettuata l'operazione, sul foglio si verificano diversi cambiamenti. La data compare nell'etichetta dei parametri nella riga 6 e nella riga 12 compare una serie di cinque date consecutive. Questi sono tutti i valori che dipendono dal valore del parametro nella posizione D5.

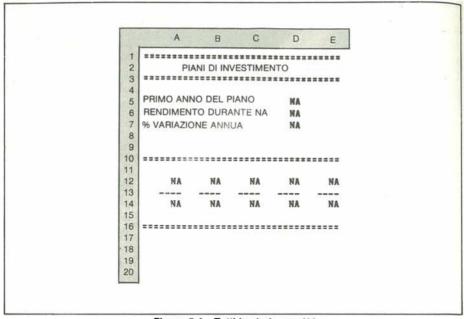



Figura 5.4 - Tutti i valori sono NA

Inseriamo adesso un valore per il rendimento del primo anno e una percentuale per la variazione. (Esprimeremo ancora una volta i valori monetari in unità di \$ 1000.)

```
>D6 23.7        {rendimento del primo anno: $ 23,700}
>D7 8.5         {variazione percentuale annua: 8.5%}
```

I valori del foglio vengono modificati dopo *ogni* immissione di valore. Il risultato finale compare nella Figura 5.5.

E' facile esaminare possibili variazioni di questo investimento. Cosa avviene se gli aumenti annui sono solo del 7.8% - quale sarebbe il rendimento nel 1987? Per saperlo, si sposti il cursore in D7, si inserisca il nuovo valore del parametro, 7.8, e si guardi il risultato. Il nuovo rendimento per l'anno 1987 è 32.01 (o \$ 32.010).

Talvolta si vorrà modificare più di un valore per volta. Per esempio, cosa succede se il rendimento del primo anno è solo di \$19.500 e l'aumento annuo è del 9%? Quando si inserisce ognuno di questi nuovi valori nella tavola dei parametri, si nota come la tavola viene ricalcolata dopo *ogni* inserimento. A noi però interessa che venga effettuato solo un calcolo, dopo che sono stati inseriti *tutti e due* i valori.

Su questo particolare foglio la differenza di tempo fra l'effettuare il calcolo una volta sola o due è piccola, ma si immagini un foglio più complicato, con una tavola dei parametri più ampia e formule più complesse. Può darsi che si debba aspettare

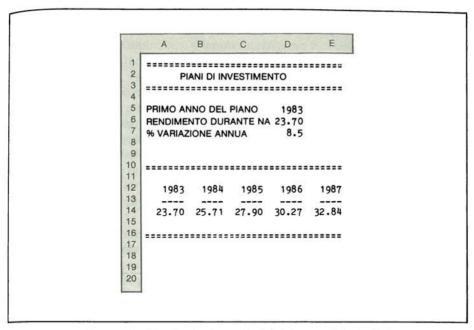



Figura 5.5 - Un primo insieme di valori dei parametri e risultati

secondi, o addirittura minuti, prima che il VisiCalc abbia effettuato i calcoli. Se nella tavola dovessero essere inseriti sei o sette valori nuovi, l'operazione sarebbe notevolmente più lenta come se il foglio venisse completamente ricalcolato sei o sette volte. Risulterebbe molto più comodo se si potesse impedire al VisiCalc di effettuare i calcoli finchè non si sono inseriti tutti i nuovi valori.

E' in queste situazioni che torna utile il comando globale di ricalcolo, /GR. Il VisiCalc ha due modalità di ricalcolo: automatica e manuale. Quando si lancia il programma, la modalità in vigore è quella automatica. Questo significa che ogni volta che viene inserito un valore, l'*intero* foglio viene ricalcolato automaticamente. Si può passare alla modalità manuale (o tornare a quella automatica) tramite il comando di ricalcolo globale.

Battendo /GR sul rigo informativo compare il messaggio:

RECALC:A M

A sta per automatico, M per manuale. La modalità corrente è l'automatica, quindi battiamo M per passare a quella manuale. Si noti come questa operazione non abbia conseguenze sullo schermo; bisogna ricordare in che modalità si è.

Modifichiamo adesso tutti e tre i valori della tavola dei parametri:

>D5 1984
>D6 35.75
>D7 8.2

I dati calcolati del foglio non subiscono cambiamenti: tutte le date e le cifre dei rendimenti annui sono quelle della situazione precedente. Questo perchè, quando si è nella modalità manuale, è necessario *richiedere esplicitamente* il ricalcolo, e cioè battere il punto esclamativo quando si vuol vedere i risultati dei nuovi valori dei parametri.

Proviamo. Battiamo e osserviamo i nuovi valori apparire tutti insieme. Il nuovo foglio elettronico dovrebbe essere come quello della Figura 5.6.

Per riassumere, abbiamo visto due strumenti per costruire fogli elettronici generali. La funzione @NA permette di distinguere il procedimento di creazione delle formule da quello dell'inserimento dei dati nelle formule. Il comando /GRM permette di controllare il procedimento di ricalcolo quando si devono modificare molti parametri: il ricalcolo viene effettuato solo quando lo si richiede battendo! Si può sempre tornare alla modalità di ricalcolo automatico impartendo il comando /GRA.)

C'è un'ultima annotazione da fare a proposito della funzione @NA. Nella tavola dei parametri possono esserci valori "facoltativi", cioè valori che compaiono in alcune situazioni ma che in altre, per qualche motivo, rimangono non disponibili. In questo caso, il fatto che un valore non sia disponibile rappresenta un'informazione significativa del foglio. In altri termini, il valore NA è un *valore* che si vorrebbe poter riconoscere nella logica del foglio elettronico.

Il VisiCalc ha ancora una volta una risposta per questa necessità. La funzione @ISNA (*is NA*, c'è NA) dà il valore vero o falso a seconda che il suo argomento sia NA o meno. Normalmente l'argomento di @ISNA sarà un riferimento a valore e la

Figura 5.6 - Nuovo calcolo indotto nella modalità di ricalcolo manuale

funzione verrà utilizzata nell'ambito di un'espressione con @IF:

@IF(@ISNA(D8),(*valore #1*),(*valore #2*))

Se @ISNA dà vero, viene scelto il valore #1; se dà falso, il valore #2.

Si possono avere situazioni in cui la non disponibilità di un dato ha un significato particolare, ed è in questi casi che @ISNA si dimostra particolarmente utile.

Il foglio dell'investimento ha una terza sezione. Nell'inserirla, impareremo alcuni aspetti del modo in cui il VisiCalc effettua i calcoli e individueremo un difetto insidioso, ma facilmente rimediabile, del nostro foglio.

IL COMANDO /GO

La riga 18 del foglio conterrà i rendimenti totali:

>A18 RENDIM. {etichetta rendimenti totali}
>B18 TOTALI

Nella posizione D18 battere la formula per trovare la somma dei rendimenti annui:

>D18 @SUM(A14.E14) {formula di addizione}

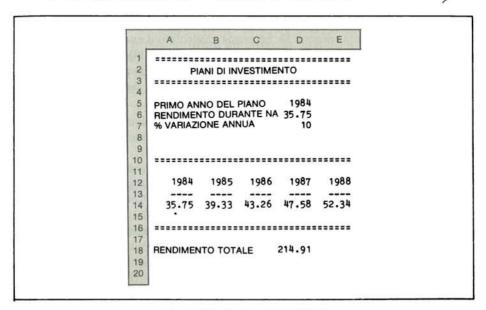



Figura 5.7 - Un errore nel calcolo

Nella posizione D18 comparirà il valore 210.57 (che significa $210 570). Guardiamo cosa succede quando modifichiamo uno dei valori della tavola dei parametri. Andiamo alla posizione D7 e cambiamo la variazione percentuale annua in 10 (cioè 10%). Battiamo! per far effettuare un calcolo. Le cifre del rendimento annuo variano tutte di conseguenzà e il totale sarà un nuovo valore. (Il risultato compare nella Figura 5.7.)

Ma c'è qualcosa che non va. La riga con i valori dei rendimenti annui non è stata sommata in modo corretto: il valore della posizione D18 dovrebbe essere 218.26, mentre è 214.91. Il VisiCalc ci ha traditi?

No, naturalmente; dobbiamo solo imparare una delle sue caratteristiche. Nel Capitolo 1, quando abbiamo esaminato per la prima volta le informazioni sullo schermo, abbiamo notato la C nell'angolo superiore destro. Questa C sta per colonna, e cioè per l'*ordine di calcolo* iniziale del VisiCalc; avevamo ignorato il messaggio perchè non erano mai sorte difficoltà.

Nella modalità di calcolo per colonne, il VisiCalc effettua i calcoli nel seguente ordine: comincia da A1, svolge i calcoli su tutti i valori della colonna A, passa poi in B1, fa i calcoli sulla colonna B e così via. Finora quest'ordine non aveva causato problemi.

Nel foglio degli investimenti, invece, abbiamo scoperto una discrepanza fra la nostra *concezione* dell'ordine di calcolo e la *realtà* del metodo del VisiCalc. Guardiamo di nuovo la Figura 5.7 per capire cosa è successo. Quando abbiamo inserito un nuovo tasso percentuale nella posizione D7, ci aspettavamo che il VisiCalc ricalcolasse tutti e cinque i valori del reddito annuo e poi trovasse la somma di queste nuove cifre per inserirla nella posizione dei guadagni totali. E' invece successo questo: il VisiCalc ha ricalcolato ogni colonna per intero, una per volta. Quando è arrivato alla colonna D, ha prima ricalcolato il nuovo introito del 1987 (la posizione D14) e poi i guadagni totali (posizione D18). A questo punto la posizione E14, nella colonna E, conteneva ancora il *vecchio* valore del 1988 (cioè quello della situazione precedente.) Quindi la cifra errata degli introiti totali era basata su quattro valori annuali nuovi e uno vecchio.

Il VisiCalc procedeva per colonne, mentre noi avevamo concepito il foglio in modo che procedesse per righe. L'incongruenza della pianificazione ha provocato l'errore.

A questo punto si sarà certamente capito il problema: talvolta è necessario essere consapevoli dell'ordine in cui il VisiCalc effettua i calcoli e progettare il foglio in modo che corrisponda a quell'ordine, oppure modificare l'ordine adeguandolo al foglio. Lo strumento per modificare l'ordine di ricalcolo è il comando /GO.

Quando si batte /GO, sul rigo esplicativo compare il messaggio

REEVAL ORDER: R C
ordine di ricalcolo: R C

Per far sì che il VisiCalc effettui i calcoli riga per riga, battere R. A questa

operazione la lettera nell'angolo superiore destro cambia in R. Il messaggio permette di conoscere sempre la modalità di calcolo del programma: R per righe, C per colonne.

Battere! Nella posizione D18 comparirà il valore corretto degli introiti totali, 218.26. Proviamo ora ad apportare altri cambiamenti ai valori dei parametri. Adesso, nella modalità di calcolo per righe, tutti i valori sono corretti.

Un approccio alternativo poteva consistere nel far effettuare i calcoli due volte, quando il programma è nella modalità per colonne. Una volta determinati i valori corretti degli introiti annui, un secondo calcolo del foglio avrebbe fornito il valore totale corretto per i cinque anni dell'investimento. Talvolta è necessario far rieffettuare i calcoli più di una volta, quando i fogli sono molto complicati. Il procedimento preferibile è il più semplice: quando si concepisce il foglio elettronico è bene decidere quale ordine di calcolo sia il migliore. Bisogna anche cercare di evitare di utilizzare troppi riferimenti a valori in posizioni "successive". Come abbiamo visto in questo esempio, i *riferimenti in avanti* possono causare problemi se non se ne è consapevoli.

L'ultimo rigo che inseriremo nel foglio è quello che conterrà il *valore attuale netto* dell'investimento. Si tratta di un valore essenziale per mettere a confronto piani di investimento diversi.

IL VALORE ATTUALE NETTO

Calcolare il valore attuale è semplicemente un modo per tener conto dell'aspetto temporale del valore del denaro. Il fatto che tale aspetto abbia un valore è una realtà quotidiana; spiega perchè si riceve un interesse quando si mette il denaro in un conto in banca o perchè si paga un interesse quando si ottiene un prestito. Nel corso del tempo il denaro può essere utilizzato per guadagnare più denaro, così le mille lire che si hanno in tasca oggi sono per noi più preziose di quelle che si riceveranno fra un anno; durante l'anno di attesa, prima di ricevere le seconde mille lire, si possono sfruttare quelle di oggi. (Possono esserci altri motivi, economici, per cui le mille lire odierne sono più preziose. Questi, però, esulano dalla presente discussione.)

Il valore attuale, quindi, rappresenta un modo per esprimere quanto vale oggi un introito futuro noto. Per esempio, supponiamo di sapere che fra cinque anni si erediteranno $ 1000. Nel frattempo, abbiamo un conto in banca sul quale otteniamo un interesse del 6% annuo. Si calcola che se oggi si mettono in banca $747, fra cinque anni si avranno $1000, pari a quanto si erediterà. Si può dire che il valore attuale dell'eredità futura, al *tasso di sconto* del 6%, è di $747.

Il *valore attuale netto* di una serie di introiti futuri è semplicemente la somma dei loro valori attuali. Il procedimento aritmetico per trovare il valore attuale netto dipende da tre fattori: il valore della somma futura, il tempo che deve passare prima di averla e il tasso di sconto. Questo calcolo può essere lungo e complicato, se effettuato a mano. La funzione @NPV (da *Net Present Value*, valore attuale netto)

lo effettua senza complicazioni. Il formato della funzione del valore attuale netto è:

```
@NPV(valore, campo)
```

L'argomento di valore rappresenta il tasso di sconto, che va espresso in forma decimale, non percentuale. L'argomento di campo rappresenta il campo dei valori futuri.

Quindi, per incorporare il valore attuale netto nel foglio elettronico, si cominci con l'aggiungere il tasso di sconto alla tavola dei parametri nella riga 8:

```
>A8 TASSO S                  {riga 8: etichetta dei nuovi parametri}
>B8 CONTO
>D8 /FG @NA                  {formato generale; valore non ancora
                             disponibile}
```

Inseriamo poi l'etichetta e la formula per il valore attuale netto:

```
>A19 VALORE                  {etichetta valore attuale netto}
>B19 ATTUALE
>C19 NETTO
>D19 @NPV(D8/100,A14.E14)    {formula valore attuale netto}
```

Si può così inserire il tasso di sconto come percentuale nella posizione D8, il valore

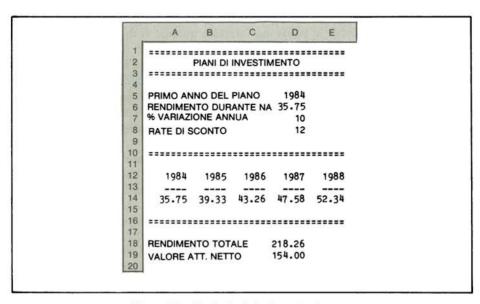



Figura 5.8 - Il calcolo del valore attuale netto

viene diviso per 100 nella formula @NPV di D19. A questo punto sia D8 che D19 contengono NA.

Supponiamo di voler effettuare i calcoli con un tasso di sconto del 12%. Portiamo il cursore in D8, inseriamo il valore 12 e battiamo per far effettuare i calcoli. Nella posizione D19 compare il valore 154.00. (Vedi Figura 5.8.) Il valore attuale di questo investimento è quindi $154.000. Si noti come questo valore sia considerabilmente inferiore agli introiti totali di $218.260.

LE FUNZIONI @ERROR E @ISERROR

Per rendere la tavola dei parametri ancor più affidabile possiamo aggiungere uno o più elementi al foglio degli investimenti. Spesso, quando si inserisce un valore in una tavola dei parametri, è importante assicurarsi che questo rientri fra un limite inferiore e uno superiore. Per esempio, nel caso di una percentuale, si può volere che il valore sia maggiore di 0 e minore di 100. Quando si inseriscono valori in una tavola dei parametri c'è sempre il rischio che si inserisca inavvertitamente un valore sbagliato, il che renderebbe i risultati non curati e inutili.

Si può utilizzare la funzione @ERROR, insieme a @IF, per stampare un messaggio di errore direttamente sul foglio se la tavola dei parametri contiene un valore inappropriato. La funzione @ERROR, che non richiede argomenti, fa apparire semplicemente la parola:

```
ERROR
errore
```

nella posizione in cui viene inserita. (Si sarà forse già notato che talvolta il VisiCalc dà questo messaggio. Di solito questo avviene quando si inserisce un comando o dati che il programma non capisce; per esempio, quando si utilizza una funzione in modo scorretto, o si scrive una formula incomprensibile, o si inserisce un procedimento aritmetico illegale, come una divisione per zero.)

Poichè probabilmente vorremo che il tasso di sconto nella posizione D8 sia maggiore di zero e minore di 100, si può ampliare la formula del valore attuale netto (D19) in modo che compaia ERROR se il tasso di sconto non rientra nel campo corretto:

```
>D19 @IF(@AND(D8<100,D8>0),@NPV(D8/100,A14.E14),@ERROR)
```

Questa funzione dice: se il tasso di sconto (in D8) è minore di 100 e maggiore di zero, calcola il valore attuale netto, altrimenti mostra ERROR in D19. Proviamo a inserire in D8 un numero all'esterno del campo corretto, per esempio 105. Battiamo per far effettuare i calcoli: in D19 comparirà ERROR.

Questo tipo di controlli sugli errori di input è una caratteristica di una buona

programmazione, tanto con il VisiCalc che con altri strumenti. I fogli elettronici saranno più affidabili se si può garantire l'accuratezza del valori dei parametri di input.

Il VisiCalc ha anche una funzione @ISERROR, simile a @ISNA; la si può utilizzare tutte le volte che è importante controllare se c'è una condizone di errore prima di passare a un calcolo. Come @ISNA dà vero o falso: vero se l'argomento del riferimento a valore contiene il valore @ERROR, falso altrimenti.

CONSIDERANDO VARIE POSSIBILITA'

Abbiamo ora davanti un foglio elettronico breve, ma completo, che si può utilizzare per mettere a confronto il valore di diversi piani quinquennali di investimento. Il valore attuale netto è lo strumento che utilizzeremo per i confronti.

Supponiamo di avere tre possibilità di investimento fra le quali scegliere, che ogni piano inizi nel 1983 e che tutti richiedano lo stesso capitale iniziale, così da poter eliminare quel valore come elemento del confronto.

- Dal piano #1 ci si può aspettare $36.500 il primo anno; per ognuno dei quattro anni successivi, il rendimento previsto *diminuirà* del 10%.

- Il piano #2 darà $16.000 il primo anno e il 32% in più ciascuno degli anni successivi.

```
      A        B        C        D        E
 1  ====================================
 2          PIANI DI INVESTIMENTO # 1
 3  ====================================
 4
 5  PRIMO ANNO DEL PIANO        1983
 6  RENDIMENTO DURANTE NA      36.50
 7  % VARIAZIONE ANNUA           -10
 8  DISCOUNT RATE (%)             12
 9
10  ====================================
11
12     1983     1984     1985     1986     1987
13     ----     ----     ----     ----     ----
14    36.50    32.85    29.57    26.61    23.95
15
16  ====================================
17
18  RENDIMENTO TOTALE         149.47
19  VALORE ATT. NETTO         110.32
20
```

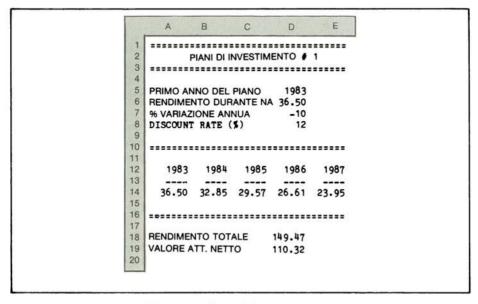



Figura 5.9 - Piano di investimento #1

```
     A       B       C       D       E
 1  ====================================
 2          PIANI DI INVESTIMENTO # 2
 3  ====================================
 4
 5  PRIMO ANNO DEL PIANO        1983
 6  RENDIMENTO DURANTE NA      16.00
 7  % VARIAZIONE ANNUA            32
 8  DISCOUNT RATE (%)             12
 9
10  ====================================
11
12     1983    1984    1985    1986    1987
13     ----    ----    ----    ----    ----
14    16.00   21.12   27.88   36.80   48.58
15
16  ====================================
17
18  RENDIMENTO TOTALE         150.37
19  VALORE ATT. NETTO         101.92
20
```

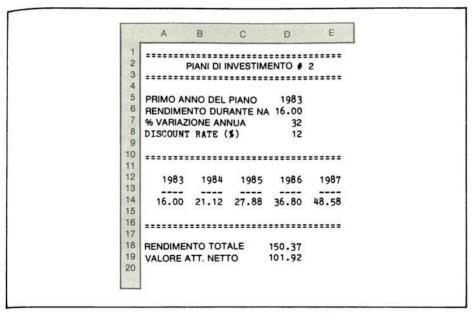



Figura 5.10 - Piano di investimento #2

```
     A       B       C       D       E
 1  ====================================
 2          PIANI DI INVESTIMENTO # 3
 3  ====================================
 4
 5  PRIMO ANNO DEL PIANO        1983
 6  RENDIMENTO DURANTE NA      30.00
 7  % VARIAZIONE ANNUA             0
 8  DISCOUNT RATE (%)             12
 9
10  ====================================
11
12     1983    1984    1985    1986    1987
13     ----    ----    ----    ----    ----
14    30.00   30.00   30.00   30.00   30.00
15
16  ====================================
17
18  RENDIMENTO TOTALE         150.00
19  VALORE ATT. NETTO         108.14
20
```

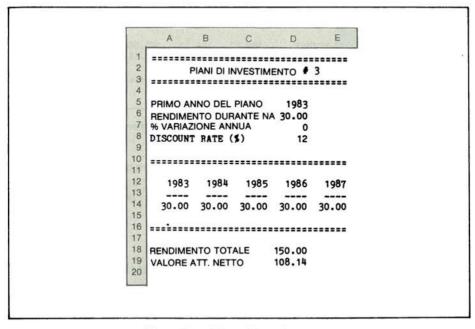



Figura 5.11 - Piano di investimento #3

- Il piano #3 darà $30.000 ciascuno dei cinque anni.

Sappiamo che il rendimento totale di ogni piano è approssimativamente di $150.000, ma vorremmo confrontare i lori valori attuali netti. Per il calcolo si userà un tasso di sconto del 12%.

Le Figure 5.9, 5.10 e 5.11 mostrano i tre piani di investimento sviluppati sul foglio elettronico. (Si noti come il titolo del foglio sia stato leggermente modificato per identificare le diverse situazioni.) In ciascun caso è stato necessario inserire solo due valori di parametri nuovi - il rendimento del primo anno e la variazione percentuale annua - dopo di che è stato fatto effettuare il calcolo. Si dovrebbe creare i tre fogli e memorizzarli su disco. Li utilizzeremo ancora nel Capitolo 6.

	A	B	C	D
1	====================================			
2	PIANI DI INVESTIMENTO			
3	====================================			
4		#1	#2	#3
5		==	==	==
6				
7	ANNO 1	36.50	16.00	30.00
8	% VARIAZ	-10	32	0
9	TASSO SC.	12	12	12
10				
11	====================================			
12				
13	TOTALE	149.47	150.37	150.00
14	V.A.N.	110.32	101.92	108.14
15				
16	====================================			

Figura 5.12 - Sommario dei piani di investimento

La Figura 5.12 mostra un riepilogo dei tre investimenti. Con un valore attuale netto di $110.320, l'investimento 1 sembra il più conveniente, anche se il rendimento totale effettivo è leggermente inferiore a quello degli altri due.

SOMMARIO

Se la struttura dei fogli elettronici viene progettata con cura, si possono costruire strumenti che continueranno a essere utili nel tempo e non verranno utilizzati una volta sola. Il principio essenziale da considerare è che i dati specifici tendono a variare da una situazione all'altra, mentre le formule rimangono generalmente le stesse. La *tavola dei parametri* è una struttura ideale per i fogli di utilità generale.

Il VisiCalc offre diversi strumenti particolarmente importanti in questo contesto. La funzione @NA è utile nella costruzione del foglio e permette di concentrarsi sulla scrittura delle formule senza preoccuparsi dei dati. Il comando /GRM porta il calcolo in modalità manuale, così che si possono modificare tutti i parametri che si vuole senza dover aspettare che vengano effettuati calcoli superflui dopo ciascun cambiamento. In questa modalità si stabilisce quando si vuole che vengano effettuati i calcoli battendo!
La comprensione di come il VisiCalc effettua i calcoli sul foglio è essenziale per ottenere accuratezza e affidabilità, specialmente quando la rete dei riferimenti a valori diventa complicata. Il VisiCalc può effettuare i calcoli per colonne o per righe e spetta all'utente progettare il foglio in un senso o nell'altro. Il comando /GO modifica l'ordine di calcolo. Una lettera (C o R) nell'angolo superiore destro dello schermo mostra la modalità corrente.

Un ultimo elemento di una corretta pianificazione del foglio è costituito da un controllo degli errori di input per i valori della tavola dei parametri. La funzione @ERROR può essere utilizzata per mostrare un messaggio di errore sul foglio in caso ci sia stato un errore di input.

CAPITOLO 6

I FILE DIF, PARTE I: UN'INTRODUZIONE

A COSA SERVONO I FILE DIF

Questo capitolo e i due che seguono spiegano come il programma VisiCalc utilizza i file di dati; inoltre, vengono dettagliatamente analizzati tre programmi in BASIC, quindi, a questo punto, chi non sa nulla di questo linguaggio di programmazione dovrebbe leggere l'Appendice B, "Uno sguardo al BASIC."

Un file di dati è una sequenza ordinata di informazioni permanenti, memorizzate su un supporto esterno, di solito un disco o il nastro di una cassetta. Idealmente, l'organizzazione di questo file dovrebbe permettere ai programmi che devono accedere ai suoi dati di leggerlo e comprenderlo senza difficoltà. In termini di accesso ai dati, i file di dati si dividono in due categorie: file *sequenziali*, che devono essere letti (o scritti) secondo un ordine preciso - partendo dall'inizio e procedendo dato per dato e file *ad accesso casuale*, che consentono la lettura e la scrittura dei dati in qualsiasi ordine. In questi ultimi tre capitoli parleremo di file sequenziali.

I due tipi di file possono essere rispettivamente paragonati a un libro giallo e a un dizionario: uno lo si legge dall'inizio alla fine, senza mai uscire dall'ordine sequenziale; l'altro lo si consulta in ordine casuale, a seconda delle necessità.

Il fine di un file di dati è di fornire un supporto *indipendente* per la memorizzazione delle informazioni. Indipendente da due cose: primo, indipendente da quanto succede sul computer; secondo, indipendente dai programmi che creano e utilizzano le informazioni sul file. Poichè un file di dati è immagazzinato su un supporto di memoria esterno, i dati sono al sicuro anche quando il computer viene spento. Se oggi si usa un programma per creare un file, domani vi si può tornare, sicuri di ritrovarvi le informazioni immessevi. Un file di dati può essere utilizzato da più programmi con funzioni distinte, e può addirittura essere letto da programmi di *tipo* diverso, il che significa che programmi scritti in linguaggi diversi possono scambiarsi informazioni. Perchè questo sia possibile, il file dev'essere organizzato in un formato attentamente definito che tutti i programmi che utilizzano il file devono "conoscere".

Il *Data Interchange Format* (DIF, formato per l'interscambio dei dati), è uno di questi formati. E' stato concepito per la memorizzazione dei dati del VisiCalc e per un'utilizzazione di tali dati anche da parte di altri programmi. Il VisiCalc può sia

leggere che creare file in formato DIF e il formato può essere letto (o scritto) senza difficoltà anche da programmi in BASIC.

I file DIF contengono due diversi tipi di informazioni. Primo, ci sono i dati veri e propri per la memorizzazione dei quali il file è stato creato. Secondo, ogni file DIF è *autoesplicativo*, contiene cioè informazioni su se stesso. Per esempio, una sezione del file DIF comunica quanti sono i dati che il file contiene e come sono organizzati. Inoltre, questi file dispongono di un sistema per comunicare il *tipo* - numerico o non numerico - di ciascun dato. Tutte queste informazioni sono scritte in un formato semplice, affidabile e perfettamente prevedibile.

E' importante tener presente la differenza fra un file DIF e uno di un foglio elettronico del VisiCalc. Un file di foglio elettronico contiene tutte le informazioni che si inseriscono nel VisiCalc per creare quel determinato foglio. Queste informazioni comprendono i formati, le formule e i comandi globali, oltre alle etichette e ai valori. In altri termini, un file di foglio elettronico contiene tutto quello che il VisiCalc deve sapere per duplicare esattamente quel foglio. Probabilmente, studiando questo libro, si sono già creati molti file di fogli elettronici tramite il comando /SS. Li abbiamo contraddistinti con il suffisso ".VC" perchè fosse chiaro di che tipo di file si trattava.

I file di foglio elettronico del VisiCalc sono esclusivi del VisiCalc, non sono cioè concepiti per essere utilizzati da programmi di altro genere. Un file DIF, invece, contiene solo le informazioni che probabilmente si vorrà trasferire dal VisiCalc a, per esempio, un programma in BASIC. Non si vorrà certo usare le formule o i comandi del VisiCalc con il BASIC, visto che gli sarebbero incomprensibili, mentre si può voler trasferire a un programma in BASIC i *dati*: etichette e valori di un foglio elettronico. Supponiamo di star scrivendo un programma in BASIC per elaborare i dati in un modo che nel VisiCalc è scomodo, o impossibile, ma che la fonte di tali dati, e la loro destinazione finale, sia un foglio elettronico del VisiCalc, (nei capitoli successivi vedremo programmi di questo genere): il mezzo per trasportare i dati dal foglio del VisiCalc al programma in BASIC e poi riportarli indietro è un file DIF.

Quindi un file DIF contiene i *dati* di un foglio del VisiCalc, ma non le formule o i comandi utilizzati per crearli. I file DIF, come presto vedremo, vengono creati dal comando /S#.

In questi ultimi tre capitoli ci concentreremo sull'uso dei file DIF per il trasferimento dei dati dei fogli elettronici fra il VisiCalc e il BASIC. Questi file hanno però un'altra importante utilizzazione: permettono di spostare righe e colonne di dati (di nuovo, *non* le formule o i formati) da un foglio elettronico a un altro, all'interno del VisiCalc. Per questa seconda utilizzazione *non* è necessario avere nozioni sul formato DIF: il VisiCalc pensa automaticamente a tutti i dettagli del trasferimento.

L'ultimo foglio elettronico del Capitolo 5, il sommario delle tre possibilità di investimento (Figura 5.12) - è stato creato utilizzando i file DIF per trasferire i dati dai vari fogli (Figura 5.9, 5.10 e 5.11). Prima di cominciare a parlare della struttura dei file DIF e della loro utilizzazione nei programmi in BASIC, esaminiamo questa più semplice, ma preziosissima, utilizzazione del DIF.

TRASFERIMENTO DEI DATI FRA I FOGLI ELETTRONICI

Se si sono memorizzati su disco i tre piani di investimento, si possono compiere le operazioni necessarie per la creazione del foglio elettronico di riepilogo. Si ricorderà che i valori dei parametri (data, rendimento annuo, variazione percentuale e tasso di sconto) e i totali dei calcoli (rendimenti totali e valore attuale netto) apparivano tutti nella colonna D dei tre fogli elettronici. Questa disposizione aveva permesso di creare un foglio di riepilogo trasferendo una sola colonna da ciascuno dei tre fogli.

Cominceremo dal piano di investimento #1 (Figura 5.9). Con il comando /SL carichiamo il foglio elettronico dal disco sullo schermo. Spostiamo il cursore in D5, la posizione che contiene la data del primo anno dell'investimento e che è anche quella più in alto nella colonna dei valori che si vuol trasferire. Battiamo adesso il comando di memorizzazione dei file DIF, /S#. Il rigo esplicativo mostra:

DATA: SAVE LOAD
dati: memorizza carica

A questa richiesta si risponde con una S o con una L, a seconda che si voglia memorizzare (*save*) i dati come file DIF, o caricarli (*load*) da un file DIF. In questo esempio, battiamo S, per memorizzare. Il messaggio successivo è:

DATA SAVE: FILE FOR SAVING
memorizzazione di dati: file per la memorizzazione

A questo punto, al file DIF che si sta per creare va attribuito un nome. Diamogli:

INV1.DIF

per "investimento #1." Si noti il suffisso ".DIF". E' importante. Un disco può finire col contenere molti file ed è difficile distinguerli gli uni dagli altri, se non hanno nomi che ne identificano chiaramente il contenuto e il tipo. Come abbiamo usato ".VC" per indicare un file di un foglio elettronico del VisiCalc, useremo sempre ".DIF" per indicare file DIF. (Alcune versioni del VisiCalc forniscono automaticamente questo suffisso a seguito del comando /S#; per le altre versioni è necessario batterlo.)

Fornito il nome del file, il drive rimane per un attimo in azione, poi sullo schermo appare un altro messaggio:

DATA SAVE: LOWER RIGHT
memorizzazione dati: inferiore destro

Un file DIF può contenere qualsiasi porzione rettangolare del foglio - da una sola posizione a molte righe o molte colonne. Quando si utilizza il comando /S#, il

rettangolo di dati viene definito come segue: l'angolo *superiore sinistro* è costituito dalla posizione del cursore al momento in cui viene impartito il comando (in questo caso D5). Quello *inferiore destro* viene fornito quando il VisiCalc lo richiede. Come al solito, si può battere l'indirizzo della posizione o si può portare il cursore nella posizione desiderata e premere return.

Nell'esempio, poichè il "rettangolo" è costituito da una sola colonna, il suo angolo inferiore destro è la posizione D19. Si inserisca questo indirizzo. Comparirà un ultimo messaggio al quale bisogna rispondere prima della creazione del file DIF:

DATA SAVE: R, C OR RETURN
memorizzazione dati: R, C o return

Viene chiesto se si vuol memorizzare i dati per righe o per colonne. (R o return effettua la memorizzazione per righe, C per colonne.) Quando si memorizzano i file DIF per trasferire dati su un altro foglio, non c'è molta differenza fra le righe e le colonne, *purchè si ricordi cosa si è scelto.* (La scelta sarà più importante in seguito, quando prepareremo i file DIF per programmi in BASIC.) Per ora, battiamo C per memorizzare i dati come una colonna. Fatto questo, il drive entrerà in funzione, memorizzando il file.

Questa sequenza di istruzioni va seguita per ognuno degli altri due piani di investimento. Chiameremo il secondo file DIF

INV2.DIF

e il terzo

INV3.DIF

Per facilitare l'operazione, ecco un riepilogo dei comandi da impartire per costruire i file. Innanzi tutto, cancellare lo schermo e caricare il file degli investimenti del VisiCalc. Poi dare queste istruzioni:

>D5	{porta il cursore in D5}
/S#	{richiama il comando di memorizzazione DIF}
S	{per **memorizzare** un file DIF}
INV2.DIF (return)	{il nome del file}
D19 (return)	{angolo inferiore destro del rettangolo}
C	{per memorizzare per **colonne**}

Dopo aver creato in questo modo i tre file DIF, possiamo costruire il foglio

elettronico di riepilogo. Cancelliamo lo schermo. Con il cursore nella posizione A1, battiamo i seguenti comandi:

/S#	{richiama il comando di memorizzazione DIF}
L	{per **caricare** un file DIF}
INV1.DIF (return)	{nome del file}
C	{per caricare per **colonne**}

Poichè i file DIF sono stati memorizzati per colonne, vanno ricaricati per colonne, se si vuole che abbiano la stessa disposizione nel foglio di riepilogo. Con gli stessi comandi, carichiamo INV2.DIF nella colonna B e INV3.DIF nella colonna C. Il foglio elettronico che ne risulta è quello della Figura 6.1.

L'aspetto iniziale di questo foglio elettronico è insolito, tanto che sarà forse difficile rendersi conto che si tratta della base di quello di riepilogo, ma naturalmente ormai sappiamo che bastano poche operazioni per trasformarlo. Cominceremo col cancellare le righe di dati superflui, usando /DR. Per esempio, il rendimento dell'investimento nel 1986, che compariva nella colonna D di ognuno dei fogli originali, nel foglio di riepilogo non è necessario. Dopo aver cancellato le righe indesiderate, modificheremo rapidamente il formato dei dati (/GF$; e poi /FI per le date e le percentuali). Infine inseriremo righe e colonne per mettervi un titolo; etichette per le colonne e le righe, e righe nella parte superiore dello schermo per rendere più gradevole il prospetto. Queste operazioni porteranno al foglio che si è visto alla fine del Capitolo 5.

	A	B	C
1	1983	1983	1983
2	36.5	16	30
3	-10	32	0
4	12	12	12
5			
6	=	=	=
7			
8	1986	1986	1986
9	====	====	====
10	26.6085	36.79949	30
11			
12	=	=	=
13			
14	149.4712	150.3732	150
15	110.3196	101.9154	108.1433

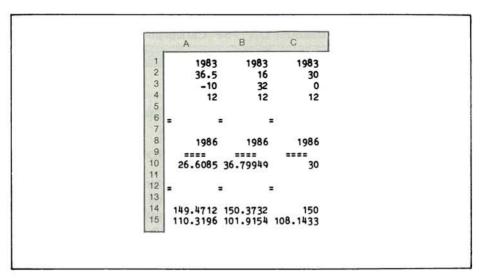



Figura 6.1 - Caricamento di file DIF su un foglio elettronico

Ma prima di apportare queste modifiche, soffermiamoci un momento a esaminare il foglio nella sua forma nuda, come viene ricostituito ricaricando il file DIF. Si può imparare molto sui file DIF esaminando i dati che questi restituiscono al foglio.

Innanzi tutto, spostiamo il cursore osservando il rigo dei contenuti, nella parte superiore dello schermo: non vi compaiono mai le *formule* che erano state utilizzate per creare i dati. Questa naturalmente è una conferma di quanto avevamo detto sui file DIF: memorizzano dati, non formule. Quando si crea un file DIF, vi viene incluso solo il *contenuto attuale* del rettangolo di foglio prescelto. Le formule che hanno creato quei contenuti non hanno alcun interesse per il DIF.

Successivamente, notiamo il formato dei dati numerici. Quello dei valori è *generale*: tutti i valori compaiono sul rigo dei contenuti con diverse cifre decimali, mentre sui fogli originali avevano il formato in dollari e centesimi. Il DIF non registra il formato originale, ma memorizza tutti i valori numerici in formato *generale*.

Infine, guardiamo cosa è successo alle linee nelle righe 6 e 12. Si ricorderà che erano state create tramite il comando di ripetizione dell'etichetta, /-. Il DIF non è in grado di registrare i *risultati* di questo comando e memorizza solo il carattere (o i caratteri) designati per la ripetizione. In questo caso il carattere è il segno uguale.

Adesso, come esercizio, trasformiamo il foglio elettronico con i soli dati in un foglio di riepilogo presentabile. Cancelliamo le righe da 6 a 12, modifichiamo il formato dei dati e inseriamo le etichette e le linee. Il tutto non dovrebbe richiedere che pochi minuti.

Un ultimo aspetto del trasferimento dei dati da un foglio all'altro: se vogliamo trasformare una riga di dati in una colonna, lo si può fare tramite un file DIF. Il metodo è semplice: basta memorizzare i dati come colonna e ricaricarli come riga, o viceversa. Proviamo a farlo con il file INV1.DIF: cancelliamo lo schermo e battiamo il comando di memorizzazione DIF, /S#. Carichiamo il file, ma in risposta al messaggio

```
DATA LOAD: R, C OR RETURN
```

battiamo R al posto di C. Sullo schermo i dati appariranno su una riga.

IL FOGLIO ELETTRONICO DELLE PRECIPITAZIONI

Nel resto di questo capitolo e nei Capitoli 7 e 8 utilizzeremo un solo file DIF. Si ricorderà come nel Capitolo 1 si sia cominciato da una tabella di numeri senza etichette, per la quale si sono immaginate tre possibili applicazioni. Abbiamo visto le prime due, che hanno portato al foglio della spesa e a quello dei venditori, e adesso siamo pronti per la terza, il foglio delle precipitazioni, che appare nella Figura 6.2. Questo foglio mostra le precipitazioni mensili medie di dieci città degli Stati Uniti.

Assumeremo che vadano soddisfatte le seguenti esigenze:

1. Il foglio dovrà poi essere ampliato per contenere più città; quindi tutte le operazioni effettuate devono essere indipendenti dalle dimensioni della tavola

2. Si vogliono calcolare alcuni indici statistici relativi ai dati. Per cominciare, si vuole la *varianza* e la *deviazione standard* dei valori delle precipitazioni mensili di ogni città.

3. Si vuol essere in grado di *ordinare* i dati del foglio in due modi: alfabeticamente, secondo il nome delle città (colonna A), o numericamente, secondo le precipitazioni totali (Colonna N, in ordine discendente).

PRECIPITAZIONI MENSILI NORMALI
(IN CENTIMETRI)

	A	B	C	D	E	F	G
4		GEN	FEB	MAR	APR	MAG	GIU
5		===	===	===	===	===	===
6	HONOLULU	11	6	8	4	3	1
7	LOS ANGL	8	7	6	3	0	0
8	SAN FRAN	11	8	6	4	1	0
9	DENVER	2	2	3	5	7	5
10	ST.LOUIS	5	5	8	10	10	11
11	NEW ORLN	11	12	14	11	11	12
12	CLEVELND	7	6	8	9	9	8
13	MIAMI	6	5	5	9	15	23
14	WASH DC	7	6	8	7	10	9
15	NEW YORK	7	7	9	8	9	8

	H	I	J	K	L	M	N
4	LUG	AGO	SET	OTT	NOV	DIC	TOTALI
5	===	===	===	===	===	===	======
6	2	2	2	4	8	9	59
7	0	0	1	1	5	6	36
8	0	0	1	3	6	10	50
9	5	3	3	3	2	1	39
10	9	7	7	7	6	5	91
11	17	13	14	6	10	13	144
12	4	8	7	7	7	6	85
13	18	17	22	21	7	4	152
14	10	12	8	7	7	8	99
15	9	10	8	7	10	9	102

Figura 6.2 - Il foglio delle precipitazioni

Vedremo come sia possibile soddisfare queste esigenze tramite tre operazioni principali: la memorizzazione dei dati in un file DIF; la scrittura di programmi in BASIC per leggere il file e svolgere le operazioni di elaborazione; e infine, per l'ordinamento, la creazione di un nuovo file DIF per riportare i dati sul VisiCalc. Ma prima di cominciare, occorre esaminare la natura del Data Interchange Format. Specificatamente, considereremo il file DIF del foglio delle precipitazioni per scoprire in che modo è organizzato un file DIF.

Creiamo una copia del foglio delle precipitazioni che utilizzeremo in seguito. Cominciamo caricando nel VisiCalc i dati dal file NUMERI.VC. Con il comando /GFI, rendiamo interi tutti i numeri del foglio. Inseriamo una colonna sulla sinistra (IC) e cinque righe in alto (/IR, cinque volte), e scriviamo il titolo, le etichette e le linee come si vede nella Figura 6.2.

Adesso siamo pronti a creare il file DIF per questo foglio. Portiamo il cursore in A6 (che contiene l'etichetta "HONOLULU") e battiamo i seguenti comandi:

/S#	{richiama il comando di memorizzazione DIF}
S#	{per **memorizzare** un file DIF}
PIOGGIA.DIF (return)	{nome del file}
N15 (return)	{l'angolo inferiore destro}
R	{per memorizzare i dati per **righe**}

Si noti come a questo file si sia dato nome PIOGGIA.DIF.

Allo scopo di esaminare un file DIF, possiamo scrivere un semplice programma in BASIC che ne mostri sullo schermo il contenuto. Nella prossima sezione di questo capitolo guarderemo il contenuto di PIOGGIA.DIF e nella sezione successiva esamineremo il programma in BASIC che ha creato la raffigurazione.

LA STRUTTURA DEI FILE DIF

La Figura 6.3 mostra il contenuto del file PIOGGIA.DIF. (Per motivi di spazio, la figura è suddivisa in colonne scelte arbitrariamente, ma si ricordi che i dati formano un file continuo.) Diamole uno sguardo, prima di esaminarla dettagliatamente. Si notano subito tutti i dati del foglio delle precipitazioni: prima i nomi delle città, poi i valori delle precipitazioni, presi dalle colonne del foglio originale. I valori delle precipitazioni sono però inframezzati da molti altri dati il cui significato non è immediatamente evidente: sono quelli che forniscono informazioni relative agli stessi file DIF. *Descrivono* il contenuto del file e sono stati progettati in funzione dei programmi che leggeranno il file. Alcuni valori, quando li esamineremo, sembreranno arbitrari e superflui. Questo è in parte dovuto al fatto che il formato DIF è stato progettato per essere flessibile e alcuni dati consentono di adibirlo ad altri usi. Quello della Figura 6.3 è il formato che può essere letto e scritto dal

```
TABLE
0,1
""
VECTORS
0,10
""
TUPLES
0,14
""
DATA
0,0
""
-1,0
BOT
1,0
"HONOLULU"
1,0
"LOS ANGL"
1,0
"SAN FRAN"
1,0
"DENVER"
1,0
"ST.LOUIS"
1,0
"NEW ORLN"
1,0
"CLEVELND"
1,0
"MIAMI"
1,0
"WASH DC"
1,0
"NEW YORK"
-1,0
BOT
0,11.18
V
0,7.62
V
0,11.18
V
0,1.52
V
0,4.83
V
0,11.43
V
0,6.6
V
0.5.59
V
0,6.6
V
0,6.86
V
-1,0
BOT
0,6.35
V
0,7.11
V
0,7.62
V
0,1.78
V
0,5.33
V
0,11.68
V
0,5.59
V
0,5.08
V
0,6.35
V
0,7.37
V
-1,0
BOT
0,8.13
V
0,5.59
V
0,6.35
V
0,3.05
V
0,7.62
V
0,13.97
V
0,7.87
V
0,5.33
V
0,8.38
V
0,9.4
V
-1,0
BOT
0,3.56
V
0,3.3
V
0,4.06
V
0,4.83
V
0,9.91
V
```

Figura 6.3 - PIOGGIA.DIF (continua)

```
0,10.67
V
0,8.89
V
0,9.3
V
0,7.37
V
0,8.38
V
-1,0
BOT
0,2.54
V
0,.25
V
0,1.02
V
0,6.86
V
0,9.91
V
0,10.67
V
0,8.89
V
0,15.49
V
0,9.55
V
0,8.89
V
-1,0
BOT
0,.76
V
0,0
V
0,.25
V
0,4.83
V
0,11.18
V
0,11.94
V
0,8.38
V
0,22.86
V
0,8.89
V
0,7.62
V
-1,0
BOT
0,1.52
V
0,0
V
0,0
V
0,4.57
V
0,9.4
V
0,17.02
V
0,3.89
V
0,17.53
V
0,10.41
V
0,9.4
V
-1,0
BOT
0,2.03
V
0,0
V
0,0
V
0,3.3
V
0,7.37
V
0,13.46
V
0,7.87
V
0,17.24
V
0,12.09
V
0,10.16
V
-1,0
BOT
0,1.78
V
0,.51
V
0,.51
V
```

Figura 6.3 - PIOGGIA.DIF (continua)

```
0,2.79
V
0,7.37
V
0,14.22
V
0,7.11
V
0,22.1
V
0,7.87
V
0,8.38
V
-1,0
BOT
0,3.81
V
0,.76
V
0,2.54
V
0,2.79
V
0,7.11
V
0,5.84
V
0,6.6
V
0,20.83
V
0,6.86
V
0,7.37
V
-1,0
BOT
0,7.62
V
0,5.08
V
0,5.84
V
0,2.03
V
0,6.35
V
0,9.91
V
0,7.11
V
0,6.86
V
0,7.37
V
0,9.65
V
-1,0
BOT
0,9.4
V
0,5.59
V
0,10.16
V
0,1.02
V
0,5.08
V
0,12.95
V
0,6.1
V
0,4.06
V
0,7.62
V
0,8.89
V
-1,0
BOT
0,58.68
V
0,35.81
V
0,49.53
V
0,39.37
V
0,91.46
V
0,143.76
V
0,84.9
V
0,152.27
V
0,99.36
V
0,102.37
V
-1,0
EOD
```

Figura 6.3 - PIOGGIA.DIF (fine)

programma VisiCalc; altri programmi che utilizzano il DIF possono avere esigenze diverse.

Il file DIF è costituito da molte brevi righe di testo. Alcune contengono due dati (separati da virgole), altre uno solo. Vedremo che molti dati del file fungono da *contrassegni*: alcuni separano le diverse sezioni del file, altri indicano il tipo di informazioni che seguiranno. Questi contrassegni sono come i separatori di plastica colorata di un contenitore di fogli: permettono di vedere dove finisce una sezione e ne comincia un'altra e identificano il contenuto delle varie sezioni.

Cominciamo a esaminare il file dettagliatamente. Questo è diviso in due parti principali; la prima, chiamata *intestazione* (*header*), contiene informazioni su come sono organizzati i dati. La seconda, chiamata *parte dati*, contiene i dati veri e propri.

L'intestazione del file della Figura 6.3 è costituita dai primi dodici righi di dati:

```
TABLE
0,1
""
VECTORS
0,10
""
TUPLES
0,14
""
DATA
0,1
""
```

Questa intestazione ha quattro diversi elementi, ognuno costituito da tre righi di dati. Il primo rigo di ogni elemento è un titolo descrittivo; il secondo contiene due numeri e il terzo è un valore *a stringa*. (Una stringa è semplicemente un valore non numerico, un altro termine per designare quello che il VisiCalc chiama *label* (etichetta)). Si noti come tutte e quattro le stringhe di questa intestazione siano *stringe vuote*, rappresentate da una coppia di virgolette senza niente in mezzo.

Questa sezione dell'intestazione può essere considerata sotto due punti di vista. Ricordiamo che si scriverà un programma in BASIC che dovrà soltanto *leggere* un file DIF e un altro che dovrà soltanto *scrivere* un file DIF, per riportare i dati nel programma VisiCalc. Dal punto di vista di un programma di lettura, l'intestazione contiene esattamente due elementi informativi significativi. (Fra un attimo vedremo di quali si tratta.) Tutti gli altri dati possono essere letti e dimenticati. I dati privi di interesse vengono chiamati "spazzatura" (*garbage*). Comunque, perchè il VisiCalc possa leggere correttamente un file DIF, *è necessario* che l'intestazione abbia questo formato. Quello che è spazzatura per un programma è essenziale per un

altro: quando redigeremo il programma di *scrittura* per riportare al VisiCalc i dati di un file DIF, questo andrà duplicato in tutti i suoi particolari. Quindi, anche se è essenziale saper riconoscere e duplicare la sezione di intestazione di un file DIF, solo alcune delle informazioni in essa contenute sono rilevanti ai fini dell'elaborazione che verrà effettuata in BASIC.

I due elementi informativi importanti sono contenuti negli elementi dell'intestazione chiamati VECTORS (vettori) e TUPLES (esimi). Questi comunicano esattamente quante righe e quante colonne di dati sono stati prelevati dal foglio del VisiCalc per essere memorizzati nel file DIF. I termini *vectors* e *tuples* non devono intimidire: sono solo i nomi con cui il DIF indica le righe e le colonne.

La corrispondenza specifica fra righe e vettori e fra colonne ed esimi dipende da come sono stati originariamente memorizzati i dati, quando è stato impartito il comando del VisiCalc /S#. Nel caso del foglio delle precipitazioni, il file è stato memorizzato per righe. Il risultato è che l'elemento dell'intestazione VECTORS comunica il numero delle righe e TUPLES quello delle colonne. Questo numero è il secondo valore del secondo rigo dell'elemento dell'intestazione. Per esempio:

```
VECTORS
0,10
""
```

indica che il file contiene 10 vettori, e:

```
TUPLES
0,14
""
```

indica 14 esimi.

La comprensione della corrispondenza fra le righe e le colonne del VisiCalc e i vettori e gli esimi del DIF è forse l'aspetto più difficile della lettura di un file DIF, e può darsi che il diagramma della Figura 6.4 renda la situazione più chiara.

L'ultimo elemento dell'intestazione è sempre quello con il titolo DATA, che ha sempre lo stesso formato:

```
DATA
0,0
""
```

Si tratta semplicemente di un contrassegno della fine della sezione dell'intestazione. In altri termini, comunica che sta per iniziare la sezione contenente i dati. E' un elemento necessario, perchè la sezione dell'intestazione è espandibile e per alcune applicazioni del DIF fra TUPLES e DATA possono esserci altri elementi dell'intestazione.

Nella seconda parte del file DIF, la sezione dei dati, i dati provenienti dal foglio del VisiCalc sono suddivisi in quelli che il DIF chiama *tuples* (esimi). Ogni esimo è

una sezione dei dati del file DIF. Tutti gli esimi di un file contengono lo stesso numero di dati gli insiemi di questi dati sono i vettori. Come abbiamo visto, gli esimi possono contenere sia righe che colonne del foglio del VisiCalc, a seconda di come i dati sono stati immagazzinati originariamente. Guardiamo di nuovo la Figura 6.4: possiamo vedere come la memorizzazione di un file DIF per righe significa che ogni esimo del DIF è costituito da una *colonna* di informazioni del VisiCalc. (Inversamente, la memorizzazione di un file DIF per colonne significa che un esimo conterrà una riga di informazioni.) Guardiamo ora il file DIF della Figura 6.3. Questo è stato memorizzato per righe, quindi gli esimi contengono colonne di informazioni. E' facile trovare il primo esimo: contiene la prima colonna del foglio elettronico, quella con i nomi delle città. Il numero di dati in ogni esimo è uguale al numero dei vettori nel file DIF.

La scelta se costruire, tramite il comando /S#, un file DIF per righe o per colonne comporta una differenza significativa nella sua organizzazione. Questa differenza si rifletterà nell'organizzazione del programma di lettura per una determinata applicazione.

Gli elementi della sezione dati sono tutti su due righi: il primo contiene due numeri (separati da una virgola), il secondo una stringa. Questa disposizione permette di memorizzare dati di tipo diverso e di specificarlo. La sezione dati del nostro file DIF contiene dati di tre tipi: contrassegni DIF, dati numerici e dati a stringa. Il

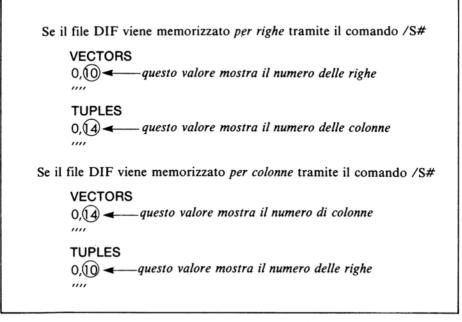



Figura 6.4 - Righe e colonne; vettori ed esimi

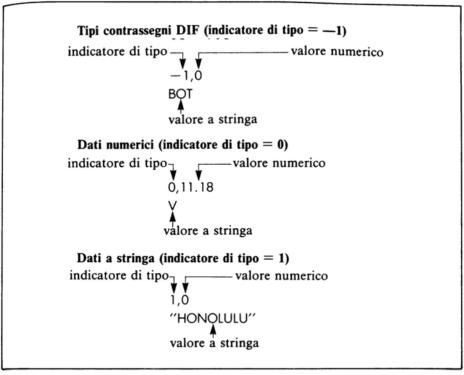



Figura 6.5 - I tre tipi di elementi della sezione dati

primo numero del primo rigo di ogni dato è l'*indicatore di tipo*, nel nostro file, gli indicatori sono -1, 0 o 1, che esamineremo uno per uno. La Figura 6.5 dà una loro visione di insieme.

Un contrassegno DIF nella sezione dati ha un indicatore di tipo -1, e ogni esimo nella sezione dati inizia con un contrassegno. Ecco qual è il suo aspetto:

```
-1,0
BOT
```

Il valore -1 indica che si tratta di un elemento di contrassegno; le lettere BOT stanno per "*beginning of tuple*" (inizio di esimo). Scorrendo il file DIF della Figura 6.3, si vede come ogni esimo (cioè, ogni colonna di dati del foglio elettronico del VisiCalc) inizi con uno di questi contrassegni su due righi. C'è anche un contrassegno per indicare la fine del file

```
-1,0
EOD
```

Le lettere EOD stanno per "*end of data*" (fine dei dati). Tanto il contrassegno BOT che quello EOD si rivelano utili in certe situazioni, come vedremo in seguito.

Gli altri due tipi di dati della sezione dati, *contengono* dati - etichette o valori - del foglio del VisiCalc. Il secondo tipo di dati è quello numerico, che è contraddistinto da un indicatore di tipo 0. Per esempio:

```
0,11.18
V
```

Il valore 0 comunica a un programma che legga il file DIF che il dato in questione è numerico. Il valore vero e proprio del dato segue direttamente l'indicatore di tipo, sullo stesso rigo - in questo caso, 11.18. La "V" del secondo rigo sta per valore. Indica semplicemente che il numero 11.18 è un valore numerico valido. Il file DIF può anche contenere i valori del VisiCalc NA e ERROR. Questi apparirebbero come segue:

```
0,0
ERROR

0,0
NA
```

Il terzo e ultimo tipo di dato della sezione dati è quello a stringa, designato dall'indicatore 1. Per esempio:

```
1,0
"HONOLULU"
```

Si noti come il valore a stringa, nel secondo rigo dell'elemento, sia fra virgolette. Per il formato DIF, le virgolette sono facoltative, e sono presenti per certe versioni del BASIC che le richiedono.

A questo punto ci si domanderà qual è la logica di questo sistema di rappresentazione dei dati, su due righi e con tre tipi. Lo scopo è semplicemente quello di dare chiare direttive a qualsiasi programma che legga un file DIF. Come l'etichetta di una lattina informa sul tipo di bevanda che ci si deve aspettare di trovarvi dentro, l'indicatore di tipo comunica al programma di lettura il *tipo* dei dati immagazzinati nell'elemento, e questo è un aspetto importante. I linguaggi di programmazione come il BASIC gestiscono la memorizzazione delle stringhe e dei valori numerici in modo diverso. Quando si scrivono istruzioni di input in un programma in BASIC, si deve specificare il tipo dei dati che il programma deve aspettarsi di ricevere, se numerici o non numerici. Il primo passo per leggere i dati di un file DIF può essere un programma in BASIC che controlli il valore dell'indicatore di tipo dei dati. Se

tale indicatore è 1, come in questa sequenza:

```
1.0
"HELLO"
```

il programma può considerare 0 come spazzatura e memorizzare la stringa "HELLO" come la parte significativa di questo dato. D'altro canto, se l'indicatore di tipo è 0, come qui:

```
0,100
V
```

il programma sa che deve leggere e memorizzare il valore numerico 100 e considerare la "V" spazzatura.

Per riassumere, abbiamo visto che un file DIF fornisce, oltre ai dati che contiene, molte informazioni importanti su se stesso:

- La sezione di intestazione indica quanti sono i dati contenuti nel file e come sono organizzati. Questa informazione è espressa in termini di VECTORS (vettori) e TUPLES (esimi). La corrispondenza fra vettori, esimi, righe e colonne dipende dal modo in cui il file è stato memorizzato.

- La sezione dati del file DIF ha *contrassegni* che indicano l'inizio degli esimi (BOT) e la fine del file (EOD).

- Ogni elemento dei dati indica, sistematicamente, il tipo di dati che contiene.

Tutte queste informazioni sono concepite in funzione dei programmi di lettura dei file. Si sarà notato come alcune di queste informazioni siano ripetitive. Per esempio, se si sa quanti sono gli esimi del file e quanti sono i valori di ogni esimo (cioè il numero dei vettori), i contrassegni BOT non sono indispensabili: si può trovare l'inizio di ogni esimo contando gli elementi dei dati.

Questa sovrabbondanza è voluta; le diverse applicazioni possono fare un uso diverso delle caratteristiche del DIF, sfruttando le più convenienti per la necessità specifica.

Il nostro primo programma in BASIC sarà molto semplice, ma illustrerà comunque come alcune delle caratteristiche del DIF rendano più semplice la lettura dei file in BASIC.

UN PROGRAMMA IN BASIC PER MOSTRARE IL CONTENUTO DEI FILE DIF

Il programma della Figura 6.6 per mostrare il contenuto di un file DIF, è il primo dei tre programmi in BASIC che esamineremo in questi ultimi capitoli. Anche chi sa poco o nulla di BASIC, dovrebbe essere in grado di *utilizzare* questi programmi sul proprio personal computer. Alcune parti di questi programmi, specificamente le subroutine per la gestione dei file, vanno modificate a seconda delle versioni del BASIC. L'Appendice A fornisce le subroutine per i BASIC di tre dei più noti personal computer; nella prossima sezione del capitolo verrà spiegato come utilizzarle.

Quando si lancia il primo programma, sullo schermo compaiono i seguenti messaggi:

```
MOSTRA    UN   FILE    DIF
======  =  ========
NOME DEL FILE? □
```

A questo punto il programma aspetta che venga fornito il nome del file DIF da esaminare. Controllare che nel drive ci sia il disco giusto e battere il nome del file: lo schermo si cancellerà e vi apparirà, rigo per rigo, il contenuto del file DIF.

Esaminiamo rapidamente la struttura di questo programma. I primi 13 righi (cioè quelli da 10 a 130) costituiscono la sezione del "programma principale". Questa sezione legge semplicemente il nome del file di cui si vuol esaminare il contenuto e poi richiama le subroutine che effettuano le operazioni di apertura, lettura e chiusura del file. (Nel BASIC, una *subroutine* è un gruppo distinto di righi di programma per lo svolgimento di un compito specifico. Una subroutine viene "richiamata" con l'istruzione GOSUB. Al termine di ogni subroutine, un'istruzione RETURN rimanda il controllo del programma al rigo che segue l'istruzione GOSUB originale.)

Il rigo 40 della sezione del programma principale cancella lo schermo con l'istruzione:

```
HOME
casa
```

Il contenuto dello schermo scompare e il cursore si porta nell'angolo in alto a sinistra. Alcune versioni del BASIC utilizzano il comando

```
CLS
```

al posto di quello HOME, quindi può darsi che questo rigo debba essere modificato. I righi 50 e 60 stampano il titolo del programma e il 70 fa saltare tre righe sullo schermo eseguendo tre volte l'istruzione PRINT:

```
10 REM *** QUESTO PROGRAMMA MOSTRA
20 REM *** SULLO SCHERMO
30 REM *** UN FILE DIF
40 HOME : REM IN ALCUNI BASIC "CLS"
50 PRINT "MOSTRA UN FILE DIF"
60 PRINT "======= = === ===="
70 PRINT : PRINT : PRINT
80 INPUT "NOME DEL FILE? ";F$
90 HOME
100 GOSUB 1000 : REM APERTURA DEL FILE PER LA LETTURA
110 GOSUB 200 : REM LETTURA E RAPPRESENTAZIONE FILE
120 GOSUB 1200 : REM CHIUSURA FILE
130 END
200 REM *** LETTURA E RAPPRESENTAZIONE FILE
210 Q$ = CHR$ (34) : REM VIRGOLETTE
220 GOSUB 2000 : REM LETTURA ELEMENTO INTESTAZIONE
230 PRINT T$
240 PRINT V1;",";V2
250 PRINT Q$; S$, Q$
260 IF T$ <> "DATA" GOTO 220
270 GOSUB 2100 : REM LETTURA ELEMENTO DATI
280 PRINT V1;",";V2
290 IF V1 = 1 THEN PRINT Q$; S$; Q$
300 IF V1 <> 1 THEN PRINT S$
310 IF S$ <> "EOD" GOTO 270
320 RETURN
1000 REM *** APERTURA FILE PER LETTURA
     **** VEDI APPENDICE A ****
1200 REM *** CHIUSURA FILE
     **** VEDI APPENDICE A ****
2000 REM *** LETTURA ELEMENTO INTESTAZIONE
     **** VEDI APPENDICE A ****
2100 REM *** LETTURA ELEMENTO DATI
     **** VEDI APPENDICE A ****
```

Figura 6.6 - Primo programma in BASIC: rappresentazione di un file DIF

```
70 PRINT:PRINT:PRINT
```

Il rigo 80 contiene l'istruzione di INPUT per il nome del file:

```
80 INPUT"NOME DEL FILE?";F
```

In alcune versioni del BASIC, l'istruzione INPUT scrive automaticamente un punto interrogativo dopo il messaggio di input. Se il BASIC che si ha a disposizione è di questo tipo, il rigo 80 dovrebbe essere modificato nel modo seguente:

```
80 INPUT"NOME DEL FILE";F$
```

Le linee 100, 110 e 120 sono richiami di subroutine:

```
100 GOSUB 1000
110 GOSUB 200
120 GOSUB 1200
```

La subroutine del rigo 1000 apre un file su disco per leggerlo. La subroutine che effettua il lavoro vero e proprio è quella del rigo 200: legge il file DIF e ne mostra il contenuto sullo schermo. La subroutine del rigo 1200 chiude il file. Nella prossima sezione verranno esaminate le subroutine di gestione dei file.

La subroutine del rigo 200 richiama altre due subroutine, una per la lettura dell'elemento dell'intestazione e una per la lettura di un elemento di dati. Queste subroutine (rispettivamente dei righi 2000 e 2100) leggono i righi del file e assegnano valori a determinate variabili. La subroutine del rigo 200 utilizza queste variabili per mostrare i dati. Guardiamo come si svolge l'operazione.

La subroutine per la lettura dell'intestazione (del rigo 2000) assegna il titolo dell'intestazione alla variabile T$; i valori numerici alle variabili V1 e V2 e il valore a stringa a S$. I righi da 220 a 250 scrivono poi sullo schermo un unico elemento di intestazione:

```
220 GOSUB 2000
230 PRINT T$
240 PRINT V1;",";V2
250 PRINT Q$; S$; Q$
```

La variabile Q$ contiene le virgolette, rappresentate dal 34 del codice ASCII:

```
210 Q$ = CHR$(R4)
```

Questa variabile viene usata per simulare il contenuto esatto degli elementi a stringa del file.

Il rigo 260 fa tornare ripetutamente al 220 per mostrare un altro elemento di intestazione, finchè l'elemento DATA, l'ultimo della sezione di intestazione, non è stato letto e mostrato:

```
260 IFT$ <> "DATA" GOTO 220
```

Questo è il primo esempio di utilizzazione di un contrassegno di un file DIF per controllare l'azione del programma.

I righi da 270 a 310 stampano la sezione dati del file. La subroutine del rigo 2100 assegna i valori numerici a V1 e V2 e il valore a stringa a S$; il rigo 280 stampa i valori numerici separati da un virgola:

```
270 GOSUB 2100
280 PRINT V1;",";V2
```

Se il valore a stringa è un'etichetta del foglio elettronico del VisiCalc, va mostrata fra virgolette. Se è un contrassegno del DIF (BOT, EOD o V) dovrebbe apparire senza virgolette. I righi 290 e 300 utilizzano l'indicatore di tipo, memorizzato nella

variabile V1, per decidere come mostrare il valore a stringa:

```
290 IF V1 = 1 THEN PRINT Q$;S$;Q$
300 IF V1 < > 1 THEN PRINT S$
```

Finalmente, il rigo 310 rimanda ripetutamente il controllo del programma al rigo 270, finchè non viene incontrato il contrassegno EOD, l'ultimo elemento del file:

```
310 IF S$ < > "EOD" GOTO 270
```

Una volta che tutto il contenuto del file è stato mostrato, il rigo 320 rimanda il controllo alla sezione del programma principale:

```
320 RETURN
```

Il difetto principale di questo programma è che non si ferma mai per consentire di esaminare il file DIF comodamente. I dati appaiono e scompaiono piuttosto velocemente, il che può essere scomodo se si deve esaminare una particolare sezione dei dati. Ci sono diversi rimedi a questo difetto, a seconda del computer e delle esigenze. Il più evidente consiste nello *stampare* i dati del file su carta, non solo di farli apparire sullo schermo. Alternativamente, su alcuni computer, si può interrompere momentaneamente l'azione del programma battendo un carattere di controllo speciale. Infine, se nessuna delle due soluzioni è soddisfacente, si può aggiungere al programma una subroutine che interrompa l'azione e la riprenda solo quando viene impartito un comando. Questa routine può essere qualcosa del genere:

```
400 INPUT "PER CONTINUARE BATTERE <RETURN>"; A$
410 RETURN
```

La routine può essere richiamata periodicamente durante la visualizzazione del file, però è utilizzabile solo se la versione del BASIC di cui si dispone consente di inserire dati dalla tastiera mentre c'è un file di dati esterno aperto.

Useremo nuovamente le quattro subroutine di questo programma per la gestione dei file (dei righi 1000, 1200, 2000 e 2100) nei programmi dei capitoli 7 e 8. Avremo poi bisogno di diverse altre subroutine per la gestione dei file. Tutte insieme formeranno una piccola "biblioteca" di routine riutilizzabili per l'input e l'output dei dati di un file DIF, e nella prossima sezione verrà esaminato il loro uso.

LE SUBROUTINE PER LA GESTIONE DEI FILE

Le otto subroutine per la gestione dei file di questa biblioteca sono molto brevi. Sfortunatamente le istruzioni per aprire, chiudere, leggere e scrivere i file esterni

Numero di rigo della subroutine	Descrizione
1000	Apre un file per *leggere* dati. Prima di richiamare la subroutine bisogna assegnare un nome di file a F$
1100	Apre un file per *scrivere* dati. Prima di richiamare la subroutine bisogna assegnare un nome di file a F$
1200	Chiude un file
2000	Legge un elemento di intestazione. Assegna i valori nel seguente modo: T$= titolo V1 = primo valore numerico V2 = secondo valore numerico S$= valore a stringa
2100	Legge un elemento dei dati. Assegna i valori nel seguente modo: V1 = indicatore di tipo V2 = valore numerico S$= valore a stringa
3000	Scrive un elemento di intestazione. Prima di richiamare questa subroutine è necessario assegnare valori nel seguente modo: T$= titolo V1 = primo valore V2 = secondo valore S$= valore a stringa
3100	Scrive un elemento dei dati *senza* mettere la stringa fra virgolette. Prima di richiamare questa subroutine, è necessario assegnare valori nel seguente modo: V1 = indicatore di tipo V2 = valore numerico S$= valore a stringa
3200	Scrive un elemento dei dati *mettendo* la stringa fra virgolette. Prima di richiamare questa subroutine, è necessario assegnare valori nel seguente modo: V1 = indicatore di tipo V2 = valore numerico S$= valore a stringa

Figura 6.7 - Subroutine per la gestione dei file

variano in modo significativo a seconda delle versioni del BASIC. I *concetti* della gestione dei file sono gli stessi, ma la sintassi dei comandi utilizzati per svolgere le operazioni cambia.

Per questo motivo, conviene sempre isolare tutte le istruzioni per la gestione dei file in routine facilmente identificabili e facilmente modificabili, ed è proprio quello che faremo nei tre programmi in BASIC del libro. La maggior parte delle altre sezioni dei programmi dovrebbero funzionare, con pochissime modifiche, su quasi tutti i BASIC. Le routine per i file, invece, vanno specificamente adattate al computer su cui si lavora.

Le otto routine di cui si avrà bisogno per questi programmi sono identificate e descritte nella Figura 6.7. Per i righi specifici di programma da battere, si deve consultare l'Appendice A, che fornisce tre diverse versioni delle routine. Se si utilizza uno dei tre computer cui si è accennato, non dovrebbero esserci problemi per includere le subroutine in un programma. Se si usa un altro computer, si dovrà vedere come la sua versione del BASIC svolge le funzioni di gestione dei file: ci sono buone probabilità che siano simili a quelle di uno dei tre elencati nell'Appendice A.

Si noti la differenza fra le routine che leggono i dati e quelle che li scrivono. Dopo aver richiamato le routine di lettura, i dati sono disponibili nelle variabili specificate. *Prima* di richiamare le routine per la scrittura, è necessario *assegnare* valori alle variabili specificate. Nel BASIC tutte le variabili sono *globali*, cioè un valore assegnato a una variabile in una routine è disponibile a tutte le routine del programma.

SOMMARIO

I file DIF possono essere creati o letti dal programma VisiCalc. Sono preziosi per lo scambio dei dati sia da un foglio elettronico a un altro che da un programma in BASIC a un foglio elettronico del VisiCalc.

La struttura dei file DIF ha lo scopo di rendere evidente al programma di lettura del file l'organizzazione interna della memorizzazione dei dati. Per questo motivo, i file DIF del VisiCalc contengono due tipi di informazioni: quelle su se stessi, e i dati del foglio elettronico veri e propri. Nel file DIF queste informazioni sono divise in due sezioni: l'intestazione e la sezione dati. L'intestazione comunica quanti sono i dati del file e, indirettamente, il numero di colonne e di righe del foglio elettronico da cui sono stati copiati. La sezione dati dispone di un elegante sistema per indicare il *tipo* di ciascun dato. Questo è essenziale per un'efficiente lettura dei dati dei programmi in BASIC.

Abbiamo visto un semplice programma in BASIC che legge e mostra, ma *non* memorizza, i dati di un file DIF. Memorizzare e utilizzare i dati dei file DIF è un po' più complicato, come vedremo nel prossimo capitolo.

CAPITOLO 7

I FILE DIF, PARTE II: LETTURA DI UN FILE DIF DAL BASIC

QUANDO NON VA USATO IL VISICALC

Uno dei principali argomenti trattati in questo capitolo, e in effetti un tema ricorrente del libro, è come utilizzare in modo efficiente i diversi strumenti di programmazione disponibili su un personal computer. Esamineremo alcuni principi generali per prendere decisioni in merito, e l'esempio specifico con cui tali principi verranno illustrati è un programma in BASIC per la lettura del file DIF dei dati delle precipitazioni (che abbiamo incontrato nel Capitolo 6) e il calcolo di alcuni semplici indici statistici. Poichè il programma ha scopo esemplificativo riguardo gli strumenti di programmazione, questo capitolo andrebbe letto e analizzato anche da chi non è interessato o non ha bisogno della statistica.

Alcune operazioni matematiche non sono particolarmente adatte al formato del foglio elettronico del VisiCalc. Questo non significa che sia impossibile svolgerle tramite questo programma, ma che altri strumenti di programmazione possono gestirle più efficientemente. In generale, tutte le volte che un'operazione comporta il trovare molti valori *intermedi* prima di arrivare al risultato finale, effettuarla con il VisiCalc può essere piuttosto complicato. Ci si può ritrovare con un foglio pieno di valori calcolati che in effetti non ci interessano. Una volta che i calcoli principali sono stati completati, i valori intermedi che hanno portato al risultato diventano irrilevanti, però rimangono lì, e occupano spazio. Quello che è peggio è che, probabilmente, calcolare questi valori intermedi con il VisiCalc richiederà un tempo esagerato. Va ricordato che il VisiCalc è progettato per un calcolo e una *presentazione* efficienti di dati numerici e non numerici. Quando ci sono molti dati intermedi non veramente utili, può darsi che usare il VisiCalc rappresenti uno spreco.

Si può però pensare che anche le alternative richiedano molto tempo. Se, per motivi di efficienza, sono necessari molti passaggi complicati, fra cui la creazione di un file DIF e la scrittura di un programma di lettura, può essere preferibile rimanere sul VisiCalc, anche se la soluzione del problema risulta un po' goffa. La risposta a questa obiezione è semplice: se un'operazione va effettuata una sola volta, la si svolgerà naturalmente nel modo più rapido possibile, senza preoccuparsi molto

dell'eleganza della soluzione. Ma nella vita le operazioni da svolgere una volta sola sono rare, e nella programmazione anche di più. Ci sono buone probabilità che l'esigenza odierna si ripresenti domani, o la settimana prossima o il mese prossimo. Il tempo speso oggi per arrivare alla migliore soluzione del problema sarà più che compensato quando si dovrà svolgere la stessa operazione in futuro.

Inoltre, la difficoltà di scrivere un programma in BASIC per gestire un file DIF diminuisce a ogni scrittura. La chiave di questa diminuzione è l'uso delle subroutine. Abbiamo già costituito una biblioteca di brevi e semplici subroutine che svolgono le operazioni di apertura, lettura e scrittura dei file. In questo capitolo aggiungeremo a quella biblioteca tre subroutine di livello più elevato che potremo riutilizzare nel programma del Capitolo 8.

Inoltre, via via che si acquista familiarità con il mondo della programmazione, si scoprono molte fonti di subroutine, per soddisfare praticamente qualsiasi necessità. Alla fine, se si organizza il lavoro con attenzione, si scopre che scrivere un programma in BASIC consiste essenzialmente nel radunare le subroutine necessarie, dalla biblioteca che si è formato o da altre fonti alle quali si ha accesso, e di metterle insieme per risolvere il problema che ci confronta.

Il foglio delle precipitazioni rappresenta un tipico esempio di una situazione in cui può essere conveniente associare il VisiCalc a un programma in BASIC per l'elaborazione dei dati. Esaminiamo i passaggi attraverso i quali questa combinazione verrà realizzata:

- Input dei soli dati tramite il VisiCalc, che, come abbiamo visto, è l'ideale per questa operazione. Quando i dati necessari per un'applicazione sono stati inseriti su foglio, possono essere controllati con comodo e, se un valore è sbagliato, lo si può modificare senza toccare gli altri.

- Completate le operazioni di input e di verifica, creazione, tramite il comando /S# del VisiCalc, di un file DIF contenente i soli dati. Questo passaggio non richiede che qualche secondo.

- Lettura del file DIF tramite un programma in BASIC. (Arrivati alla fine di questo capitolo, avremo un insieme completo di subroutine per la lettura dei dati di un file DIF e per memorizzarli in una *matrice* del BASIC.)

- Utilizzazione di una breve e semplice ruotine in BASIC per effettuare i calcoli statistici e mostrare i risultati sullo schermo.

Per riassumere, ecco quattro domande che ci si può porre su qualsiasi applicazione potenziale del VisiCalc:

1. L'operazione implica l'input accurato e affidabile di una notevole quantità di dati?

2. I calcoli che devono poi essere effettuati su tali dati richiedono formule o valori intermedi che il VisiCalc fornisce solo con difficoltà, goffamente o in modo non efficiente?

3. E' probabile che si debba effettuare l'operazione più di una volta?

4. Si può facilmente scrivere, o trovare, le subroutine in BASIC appropriate per effettuare i calcoli che creano problemi?

Se la risposta a queste domande è "sì", è probabile che l'applicazione sia l'ideale per le capacità *combinate* del VisiCalc e del BASIC.

INDICI STATISTICI DEL FOGLIO DELLE PRECIPITAZIONI

Gli indici statistici che calcoleremo per ogni città del foglio delle precipitazioni sono la *varianza* e la *deviazione standard.* Si tratta di due misure di dispersione rispetto alla media; in altri termini, descrivono come una serie di dati si distanzia dal suo valore medio.

La varianza e la deviazione standard possono essere calcolate nel seguente modo:

1. Calcolo del valore medio della serie.

2. Sottrazione di ogni valore dalla media ed elevazione al quadrato delle differenze.

3. Somma di tutti i quadrati delle differenze del passaggio 2.

4. Divisione della somma (del passaggio 3) per il numero di valori meno 1; questo valore è la varianza.

5. Estrazione della radice quadrata della varianza; questo valore è la deviazione standard.

Se queste operazioni vengono effettuate su un foglio del VisiCalc, i valori intermedi del passaggio 2 formano una tavola di numeri grande come quella dei dati originali sulle precipitazioni. In BASIC, invece, questi valori intermedi non devono essere memorizzati; piuttosto, via via che ogni valore viene calcolato, può essere addizionato in un totale parziale.

La Figura 7.1 mostra l'output del programma in BASIC che esamineremo in questo capitolo. Si tratta di un programma semplice da usare. Inizia mostrando un titolo sulla parte superiore dello schermo:

```
STATISTICHE          DA   UN   FILE   DIF
===========          ==   ==   ====   ===
```

e poi richiede di inserire il nome del file DIF con cui si vuol lavorare:

NOME DEL FILE?

A questo punto bisogna controllare che il drive contenga il disco giusto e inserire il nome PIOGGIA.DIF. Il drive entrerà in funzione e sullo schermo apparirà il messaggio

FILE IN LETTURA

Dopo alcuni secondi di lettura e di calcolo, il programma mostrerà sullo schermo una tavola su tre colonne. La prima conterrà i nomi delle città; la seconda le varianze e la terza le deviazioni standard dei dati sulle precipitazioni.

```
STATISTICHE DA UN FILE DIF
========== ==== = === ====


NOME DEL FILE? PIOGGIA.DIF


FILE IN LETTURA

CITTA'           VAR.           DEV.ST.
====             ====           =======
HONOLULU         12.25          3.5
LOS ANGL         9.26           3.04
SAN FRAN         16.42          4.05
DENVER           2.89           1.7
ST.LOUIS         4.35           2.08
NEW ORLN         7.57           2.75
CLEVELND         2.12           1.46
MIAMI            53.88          7.34
WASH DC          2.9            1.7
NEW YORK         1.09           1.05
```

Figura 7.1 - Output del programma delle statistiche del DIF

MEMORIZZAZIONE DI UN FILE DIF TRAMITE IL BASIC

Il listato del programma per il calcolo delle statistiche del file DIF è nella Figura 7.2. Le tre subroutine dei righi 300, 400 e 500 sono quelle che leggono i file DIF e memorizzano i dati del foglio in due *matrici* del BASIC. Queste tre subroutine ricompariranno nel programma di ordinamento del Capitolo 8; cominceremo da loro, poi analizzeremo il programma delle statistiche.

Una matrice è una *struttura di dati* che immagazzina tavole di valori numerici o non numerici. Tutti i dati di una matrice in BASIC devono essere dello stesso tipo ed è possibile definire sia matrici di dati numerici che matrici di dati a stringa. E' chiaro che la matrice rappresenta una struttura ideale per la memorizzazione dei dati di un foglio elettronico contenuti in un file DIF. In questo programma verranno definite due matrici, una bidimensionale per la memorizzazione delle righe e delle colonne del foglio, e una monodimensionale a stringa per la memorizzazione dei nomi delle città.

Una matrice in BASIC viene definita da un'istruzione DIM. Il rigo 90 definisce le due matrici di questo programma:

```
90 DIM T1(T,V), T1$(V)
```

Il nome della matrice numerica bidimensionale è T1; viene definita in modo da contenere fino a T colonne o V righe di dati. La matrice a stringa, T1$, conterrà fino a V valori a stringa. Poichè le variabili T e V vengono usate nella definizione delle matrici T1 e T1$ dobbiamo chiaramente assegnare valori a T e a V prima di arrivare all'istruzione DIM. Da dove verranno questi valori? Chiaramente dalla sezione di intestazione del file DIF.

Diamo un'occhiata alla subroutine che inizia al rigo 300 del programma. Questa subroutine controlla la lettura dell'intestazione e ne prende i due elementi di informazione significativi: il numero delle righe e quello delle colonne del foglio dei dati.Si ricorderà che la subroutine del rigo 2000legge un elemento dell'intestazione di tre righi:

```
310 GOSUB 2000
```

Il titolo dell'elemento viene inserito in T$ il valore che ci interessa in V2. Via via che ciascun elemento dell'intestazione viene letto, il programma esamina T$ per vedere se sia l'elemento VECTORS o quello TUPLES, nel qual caso, alla variabile appropriata (T o V) viene assegnato un valore da V2:

```
320 IF T$ = "VECTORS" THEN V = V2
330 IF T$ ="TUPLES" THEN T =V2-1
```

Poi la subroutine torna ripetutamente indietro per leggere un altro elemento dell'intestazione finchè non arriva all'elemento DATA:

```
340 IF T$ <> "DATA" THEN GOTO 310
```

Già da questa subroutine si può vedere come il programma faccia alcune ipotesi sul file DIF che sta leggendo e, indirettamente, anche sul foglio del VisiCalc. Le

```
10 REM *** PACKAGE STATISTICO PER IL DIF
20 HOME : REM ** IN ALCUNI BASIC "CLS"
30 PRINT "STATISTICHE DA UN FILE DIF"
40 PRINT "=========== ==== = === ===="
50 PRINT : PRINT : PRINT
60 INPUT "NOME DEL FILE? ";F$
65 PRINT
70 GOSUB 1000 : REM ** APERTURA FILE
80 GOSUB 300 : REM ** LETTURA INTESTAZIONE
90 DIM T1(T,V), T1$(V)
95 PRINT : PRINT "FILE IN LETTURA" : PRINT
100 GOSUB 400 : REM ** LETTURA STRINGHE
110 GOSUB 500 : REM ** LETTURA NUMERI
115 GOSUB 1200 : REM ** CHIUSURA FILE
120 PRINT "CITTA'", "VAR.", "DEV.ST."
130 PRINT "====", "====", "======"
135 DEF FNR(X) = INT(100 * X + .5)/100
140 FOR I = 1 TO V
145 PRINT T1)
150 GOSUB 3500 : REM **SUBROUTINE DELLE STATISTICHE
160 NEXT I
170 END
300 REM *** LETTURA INTESTAZIONE
310 GOSUB 2000
320 IF T $ = "VECTORS" THEN V =V2
330 IF T $ = "TUPLES" THEN T = V2 - 1
340 IF T $ = < > "DATA" THEN GOTO 310
350 RETURN
400 REM *** LETTURA STRINGHE
410 GOSUB 2100 : REM ** SPAZZATURA
420 FOR I = 1 TO V
430 GOSUB 2100 : REM ** LETTURA ELEMENTO DATI
440 T1$(I) = S$
450 NEXT I
460 RETURN
500 REM *** LETTURA NUMERI
510 FOR I = 1 TO T
520 GOSUB 2100 : REM ** BOT -- SPAZZATURA
530 FOR J = 1 TO V
540 GOSUB 2100
550 T1(I,J) = V2
560 NEXT J
570 NEXT I
580 RETURN
1000 REM *** APERTURA FILE PER LETTURA
     *** VEDI APPENDICE A ***
1200 REM *** CHIUSURA FILE
     *** VEDI APPENDICE A ***
2000 REM *** LETTURA ELEMENTO INTESTAZIONE
     *** VEDI APPENDICE A ***
2100 REM *** LETTURA ELEMENTO DATI
     *** VEDI APPENDICE A ***
3500 REM *** STATISTICHE
3510 T3 = 0
3520 FOR J = 1 TO T -1
3530 T3 = T3 + T1(J,I)
3540 NEXT J
```

Figura 7.2 - Il programma delle statistiche del DIF (continua)

```
3550 M3 = T3/(T - 1)
3560 V3 = 0
3570 FOR J = 1 TO T - 1
3580 V3 = V3 + (M3 - T1(J,I))^2
3590 NEXT J
3600 V3 = V3/(T - 2)
3610 S3 = SQR(V3)
3620 PRINT FNR(V3), FNR(S3)
3630 RETURN
```

Figura 7.2 - Il programma delle statistiche del DIF (fine)

ipotesi sono le seguenti:

1. La prima colonna del foglio dev'essere una colonna di etichette.

2. L'ultima colonna del foglio deve contenere le somme degli altri dati del foglio di ciascuna riga.

3. Il file DIF dev'essere memorizzato per righe (dal comando /S#) in modo che ogni esimo rappresenti una colonna di dati. Il primo esimo, quindi, conterrà valori a stringa e l'ultimo i totali delle righe.

Se per errore si facesse leggere al programma un file che non rispondesse a queste caratteristiche, si otterrebbero risultati imprevedibili. In questo senso, il programma è concepito per un foglio specifico; però è anche flessibile sotto un punto di vista essenziale: può contenere un numero di righe e di colonne illimitato. Quindi, se al foglio delle precipitazioni vengono aggiunte altre città, tanto questo programma che quello di ordinamento del Capitolo 8 continueranno a funzionare in modo corretto.

La subroutine del rigo 300 viene richiamata dal programma principale, subito prima dell'istruzione DIM:

```
80 GOSUB 300
90 DIM T1(T,V), T1$(V)
```

Grazie agli elementi VECTORS e TUPLES dell'intestazione, i valori V e T possono essere utilizzati per attribuire alle due matrici la lunghezza necessaria per la memorizzazione di tutti i dati del foglio.

Una volta che le due matrici sono state definite, il programma può cominciare ad assegnar loro i valori del file DIF. La subroutine del rigo 400 legge i valori a stringa del primo esimo e li assegna alla matrice T1$. La subroutine del rigo 500 legge i valori numerici dai rimanenti esimi e li assegna alla matrice T1. Entrambe le subroutine vengono richiamate dalla sezione del programma principale, subito dopo l'istruzione DIM:

```
100 GOSUB 400
110 GOSUB 500
```

La subroutine del rigo 400 inizia considerando spazzatura il contrassegno BOT:

```
410 GOSUB 2100
```

Questo significa che l'elemento del file è stato letto, ma che il programma non utilizzerà i dati che contiene. Successivamente, tramite un loop FOR, ciascun valore a stringa viene letto e memorizzato nella matrice T1$. I valori T (il numero di esimi/colonne) e V (il numero di vettori/righe) continuano a controllare l'azione del programma. Per esempio, grazie al valore V, la subroutine del rigo 400 sa esattamente quante sono le stringhe da leggere. Questi valori a stringa vengono restituiti nella variabile S$ dalla subroutine del rigo 2100:

```
420 FOR I = 1 TO V
430 GOSUB 2100
440 TI$(I) = S$
450 NEXT I
```

La subroutine del rigo 500 richiede una coppia di loop FOR *nidificati* (cioè un loop all'interno di un altro) per leggere gli esimi di dati numerici. Il loop esterno legge ciascun esimo, uno per uno:

```
510 FOR I = 1 TO T
```

e il loop interno legge ciascun valore di un dato esimo:

```
530 FOR J = 1 TO V
```

Il valore di T è di uno inferiore al numero totale degli esimi del file DIF. (Si controlli il rigo 330.) Questo perchè il primo esimo contiene stringhe, non numeri.

Il primo elemento di ogni esimo è spazzatura, il contrassegno BOT:

```
520 GOSUB 2100
```

Successivamente, nell'ambito del loop interno, il valore numerico che torna al programma nella variabile V2, viene assegnato alla matrice T1:

```
540 GOSUB 2100
550 T1(I,J) = V2
```

Per il tipo di foglio di cui ci occupiamo (cioè di uno contenente una prima colonna di stringhe e una colonna finale di totali, memorizzato per colonne in un

file DIF), le tre subroutine dei righi 300, 400 e 500, sono flessibili e affidabili per la lettura dei dati e la loro memorizzazione in matrici.

LA SUBROUTINE DELLE STATISTICHE

La subroutine che calcola la varianza e la deviazione standard dai dati delle precipitazioni di ogni città inizia al rigo 3500. La sezione del programma principale la richiama una volta per città, nell'ambito di un loop FOR:

```
140 FOR I = 1 TO V
145 PRINT T1$(I),
150 GOSUB 3500
160 NEXT I
```

Si noti come questo loop inizi stampando il nome della città, dopo di che la subroutine delle statistiche stampa i due valori statistici.

Bastano circa una dozzina di righi di programma per calcolare e stampare la varianza e la deviazione standard. Per ogni città, la subroutine delle statistiche prende un valore da ogni esimo per effettuare i calcoli. (Ogni esimo tranne l'ultimo, che contiene la colonna dei totali, che non viene utilizzata nel programma delle statistiche.) I primi cinque righi eseguibili della subroutine calcolano la media delle precipitazioni di una determinata città e la assegnano alla variabile M3:

```
3510 T3 = 0
3520 FOR J = 1 TO T-1
3530 T3 = T3 + T1(J,I)
3540 NEXT J
3550 M3 = T3/(T-1)
```

Un secondo loop FOR addiziona i quadrati delle differenze fra la media e i valori mensili:

```
3560 V3 = 0
3570 FOR J = 1 TO T-1
3580 V3 = V3 + (M3 - T1(J,I))^2
3590 NEXT J
```

Si noti che i valori intermedi, (M3 - T1(J,I))^2, non vengono memorizzati ma solo addizionati uno dopo l'altro, con il totale parziale che aumenta nella variabile V3. Vengono infine calcolate la varianza (V3) e la deviazione standard (S3):

```
3600 V3 = V3/(T-2)
3610 S3 = SQR(V3)
```

Prima di essere stampati, i valori vengono arrotondati al centesimo dalla funzione FNR, definita dall'utente:

```
3620 PRINT FNR(V3), FNR(S3)
```

Questa funzione viene definita nella sezione del programma principale, nel rigo 135:

```
135 DEF FNR(X) = INT(100 * X + .5)/100
```

Si sarà forse notato che questa funzione è molto simile a quella di arrotondamento sul foglio elettronico del Capitolo 3. I funzionamenti del VisiCalc e del BASIC possono sembrare notevolmente diversi, ma in effetti hanno molte caratteristiche simili.

SOMMARIO

Talvolta si ha a che fare con lunghe elaborazioni di dati in cui le formule matematiche sono troppo complesse perchè il VisiCalc riesca a gestirle comodamente. In questi casi conviene considerare la possibilità di utilizzare il VisiCalc solo per l'immissione dei dati e di trasferire questi ultimi a un programma in BASIC tramite un file DIF. Ci sono diversi fattori di cui va tenuto conto per decidere se valga o no la pena di scrivere un programma in BASIC per l'elaborazione dei dati del VisiCalc. Due di questi sono: il numero delle volte che si ritiene di dover utilizzare il programma e la difficoltà di scrittura (o di reperimento) delle subruotine in BASIC necessarie per costruire il programma. Alla fine, la decisione può essere in gran parte presa a seconda dei gusti.

Abbiamo esaminato un programma che effettua alcuni calcoli statistici sui valori di un file DIF. Le subroutine riutilizzabili di questo programma, che aggiungeremo alla nostra "biblioteca" di subroutine per la gestione di file DIF, sono le tre che leggono i dati di un file e li assegnano alle variabili di due matrici del BASIC.La corrispondenza fra queste matrici e la struttura del foglio elettronico del VisiCalc è notevole. Le tre subroutine si aspettano un foglio elettronico con una prima colonna di stringhe e un'ultima colonna di totali, e il file DIF corrispondente dev'essere memorizzato per righe. Le tre subroutine, però, non pongono un limite al numero delle righe o delle colonne del foglio.

Nel Capitolo 8 compieremo un'ulteriore passo per quanto riguarda la programmazione in BASIC relativa ai file DIF: impareremo a scrivere un nuovo file DIF per riportare i dati al VisiCalc.

CAPITOLO 8

FILE DIF, PARTE III: SCRITTURA DI UN FILE DIF TRAMITE IL BASIC

RICONDUZIONE DEI DATI AL VISICALC

Una volta che si è cominciato a usare programmi in BASIC per elaborazioni particolari dei dati dei fogli elettronici del VisiCalc, si arriva inevitabilmente a un punto in cui si vuol riportare al VisiCalc dati che sono nuovi o frutto di elaborazioni. Nella maggior parte dei casi, presentare i dati in formato tabulare sarà molto più semplice tramite il VisiCalc piuttosto che con il BASIC. Naturalmente è facile scrivere una subroutine in BASIC per costruire una piccola tabella di numeri, come abbiamo fatto nel programma delle statistiche del Capitolo 7, ma una volta che questa tavola compare sullo schermo, è completamente statica e per modificarla è necessario modificare e rilanciare il programma in BASIC che l'ha creata. Con il VisiCalc, invece, si può cambiare il formato dei dati, aggiungere righe e colonne, assegnare nuovi valori ai parametri e apportare istantaneamente tutte le variazioni desiderate.

Quindi, per molte applicazioni, conviene aggiungere un ulteriore passaggio al procedimento di associazione del BASIC al VisiCalc: la possibilità di riportare file DIF al VisiCalc da un programma in BASIC. In questo capitolo analizzeremo un procedimento completo. Il programma di ordinamento dell'esempio può dimostrarsi uno strumento prezioso in molte applicazioni di fogli elettronici. Ordinare è un'operazione decisamente poco pratica per il VisiCalc e invece piuttosto semplice in BASIC.

L'operazione di ordinamento di questo capitolo richiede che venga seguita una sequenza di passaggi specifica:

1. Inserimento dei soli dati in un foglio del VisiCalc, con una colonna di etichette sulla sinistra e una di totali sulla destra.

2. Memorizzazione dei dati in un file DIF tramite il comando /S#. Il file va memorizzato per righe.

3. Lancio del programma di ordinamento in BASIC. (Vedremo come nella prossima sezione del capitolo.) Il programma di ordinamento creerà un nuovo file DIF con i dati originali disposti in ordine alfabetico o numerico.

4. Ritorno al VisiCalc e caricamento del file DIF ordinato nel foglio elettronico.

Vedremo come scrivere un file DIF tramite il BASIC sia solo poco più complicato del leggerne uno. E' necessaria un'analoga comprensione del formato DIF e un nuovo insieme di subroutine per la gestione dei file. Una volta che queste sono state scritte, possono essere riutilizzate per molte applicazioni.

UN PROGRAMMA DI ORDINAMENTO PER FILE DIF

La Figura 8.1 mostra il dialogo completo generato dal programma di ordinamento, la Figura 8.2 contiene il listato del programma. Il programma inizia scrivendo un titolo, poi chiede che venga inserito il nome del file DIF da ordinare:

```
ORDINAMENTO     DI UN   FILE   DIF
=========       ===     ===    ===
NOME DEL FILE
```

```
ORDINAMENTO     FILE     DIF
=======         ===      ===

NOME DEL FILE? PIOGGIA.DIF


FILE IN LETTURA

ORDINAMENTO A)LFA O N)UMERICO? A

FILE IN ORDINAMENTO

FILE IN SCRITTURA

  ********************

FILE APIOGGIA.DIF MEMORIZZATO
```

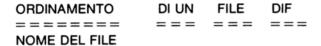
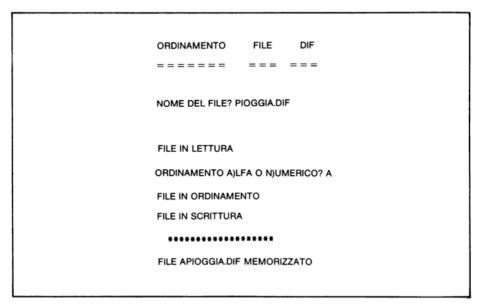



Figura 8.1 - Dialogo dal programma di ordinamento di un file DIF

```
10 REM *** PROGRAMMA PER ORDINARE UN FILE DIF
20 HOME : REM ** IN ALCUNI BASIC "CLS"
30 PRINT "ORDINAMENTO DI UN FILE DIF"
40 PRINT "======================="
50 PRINT : PRINT : PRINT
60 INPUT "NOME DEL FILE? ";F$
70 PRINT : PRINT
80 GOSUB 1000 : REM ** APERTURA FILE PER LA LETTURA
90 GOSUB 300 : REM ** LETTURA INTESTAZIONE
100 DIM T1(T,V), T1$(V)
105 PRINT : PRINT "FILE IN LETTURA" : PRINT
110 GOSUB 400 : REM ** LETTURA STRINGHE
120 GOSUB 500 : REM ** LETTURA NUMERI
130 GOSUB 1200 : REM ** CHIUSURA FILE
140 INPUT "ORDINAMENTO A)LFA O N)UMERICO? ";A$
150 IF A$ < > "A" AND A$ < > "N" THEN 140
160 F$ = A$ + F$
170 PRINT : PRINT "FILE IN ORDINAMENTO"
180 GOSUB 4000 : REM ** ORDINAMENTO FILE
190 PRINT : PRINT "FILE IN SCRITTURA"
200 GOSUB 1100 : REM ** APERTURA PER SCRITTURA
210 GOSUB 600 : REM ** CREAZIONE NUOVO FILE
220 GOSUB 1200 : REM ** CHIUSURA FILE
230 PRINT : PRINT "********************"
240 PRINT : PRINT "FILE ";F$ "MEMORIZZATO"
250 END
300 REM *** LETTURA INTESTAZIONE
310 GOSUB 2000
320 IF T$ = "VECTORS" THEN V = V2
330 IF T$ = "TUPLES" THEN T = V2 - 1
340 IF T$ < > "DATA" THEN 310
350 RETURN
400 REM *** LETTURA STRINGHE
410 GOSUB 2100 : REM ** BOT --- SPAZZATURA
420 FOR I = 1 TO V
430 GOSUB 2100 : REM ** LETTURA ELEMENTO DATI
440 T1$(I) = S$
450 NEXT I
460 RETURN
500 REM *** LETTURA NUMERI
510 FOR I = 1 TO T
520 GOSUB 2100 : REM ** BOT --- SPAZZATURA
530 FOR J = 1 TO V
540 GOSUB 2100 : REM ** ELEMENTO DATI
550 T1(I,J) = V2
560 NEXT J
570 NEXT I
580 RETURN
600 REM *** SCRITTURA NUOVO FILE
610 REM *** SEZIONE INTESTAZIONE
620 V1 = 0 : S$ = ""
630 T$ = "TABLE" : V2 = 1 : GOSUB 3000
640 T$ = "VECTORS" : V2 = V : GOSUB 3000
650 T$ = " TUPLES" : V2 = T + 1 : GOSUB 3000
660 T$ = "DATA" : V2 = 0 : GOSUB 3000
670 REM *** ESIMO A STRINGA
680 GOSUB 900 : REM ** CONTRASSEGNA BOT
```

Figura 8.2 - Il programma di ordinamento di un file DIF (continua)

```
690 V1 = 1 : V2 = 0
700 FOR I = 1 TO V
710 S$ = T1$(I): GOSUB 3200
720 NEXT I
730 REM *** ESIMI NUMERICI
740 FOR I = 1 TO T
750 GOSUB 900 : REM ** CONTRASSEGNA BOT
755 V1 = 0 : S$ = "V"
760 FOR J = 1 TO V
770 V2 = T1(I,J) : GOSUB 3100
780 NEXT J
790 NEXT I
800 V1 = -1 : V2 = 0 : S$ = "EOD"
810 GOSUB 3100 : REM ** CONTRASSEGNA EOD
820 RETURN
900 REM *** SCRITTURA CONTRASSEGNO BOT
910 V1 = -1 : V2 = 0 : S$ = "BOT" : GOSUB 3100
920 RETURN
1000 REM *** APERTURA FILE PER LETTURA
     *** VEDI APPENDICE A ***
1100 REM *** APERTURA FILE PER SCRITTURA
     *** VEDI APPENDICE A ***
1200 REM *** CHIUSURA FILE
     *** VEDI APPENDICE A ***
2000 REM *** LETTURA ELEMENTO INTESTAZIONE
     *** VEDI APPENDICE A ***
2100 REM *** LETTURA ELEMENTO DATI
     *** VEDI APPENDICE A ***
3000 REM *** SCRITTURA ELEMENTO INTESTAZIONE
     *** VEDI APPENDICE A ***
3100 REM *** SCRITTURA ELEMENTO DATI SENZA VIRGOLETTE
     *** VEDI APPENDICE A ***
3200 REM *** SCRITTURA ELEMENTO DATI CON VIRGOLETTE
   *** VEDI APPENDICE A ***
4000 REM *** ORDINAMENTO DEL FILE
4010 REM *** O IN BASE AL PRIMO ESIMO
4020 REM *** O ALL'ULTIMO
4030 FOR I = 1 TO V - 1
4040 FOR J = I + 1 TO V
4050 IF A$ = "A" THEN GOTO 4090
4060 REM ** INVERSIONE NUMERICA
4070 IF T1(T,I) < T1(T,J) THEN GOSUB 4500
4080 GOTO 4110
4090 REM ** INVERSIONE ALFABETICA
4100 IF T1$(I) > T1$ (J) THEN GOSUB 4500
4110 NEXT J
4120 NEXT I
4130 RETURN
4500 REM *** ROUTINE DI INVERSIONE
4510 H$ = T1$(I)
4520 T1$(I) = T1$(J)
4530 T1$(J) = H$
4540 FOR K = 1 TO T
4550 H = T1(K,I)
4560 T1(K,I) = T1(K,J)
4570 T1(K,J) = H
4580 NEXT K
4590 RETURN
```

Figura 8.2 - Il programma di ordinamento di un file DIF (fine)

Controllare che nel drive ci sia il disco giusto, poi inserire il nome del file da ordinare. Nell'esempio, il nome del file è PIOGGIA.DIF.

Il programma di ordinamento non invia dati allo schermo: il suo output è un nuovo file DIF, memorizzato su disco. Sullo schermo appaiono però diversi messaggi che comunicano qual è l'operazione in corso. Il primo messaggio è:

FILE IN LETTURA

che informa del fatto che il programma ha trovato sul disco il file PIOGGIA.DIF e sta leggendone e memorizzandone i dati nella memoria attiva del computer. Il messaggio successivo è un'altra richiesta:

ORDINAMENTO A(LFA O N(UMERICO?

In risposta a questo messaggio si deve battere "A" o "N" per indicare come si vuole che i dati vengano memorizzati. Il programma disporrà i dati in ordine alfabetico, secondo la prima colonna di etichette, o numerico, secondo l'ultima colonna di valori totali. Cominceremo disponendo il file in ordine alfabetico, come si può vedere nel dialogo dell'esempio.

Dopo aver risposto alla richiesta, comparirà il messaggio:

FILE IN ORDINAMENTO

Il processo di ordinamento per dieci righe di dati soltanto, non richiede molto tempo, ma se in seguito verranno aggiunte altre città, sarà più lungo. Quando l'operazione è conclusa, compare un altro messaggio:

FILE IN SCRITTURA

e contemporaneamente il drive entra in funzione, mentre il programma memorizza il nuovo file DIF sul disco. Quando l'operazione è terminata, sullo schermo appare un ultimo messaggio:

****
FILE APIOGGIA.DIF MEMORIZZATO

Questo messaggio comunica alcuni fatti importanti sull'azione del programma di ordinamento. Questi *non* distrugge il file DIF originale, non ordinato, PIOGGIA-.DIF, ma ne crea uno *nuovo* per la versione ordinata di PIOGGIA.DIF. Il programma dispone di un sistema per attribuire un nome facilmente riconoscibile al nuovo file che crea: se si richiede un ordinamento alfabetico, viene aggiunta una "A" all'inizio del nome originale; se invece l'ordinamento richiesto è numerico, il nome del nuovo file comincerà con una "N". Quindi il nome del file DIF delle precipita-

zioni, ordinato alfabeticamente, è APIOGGIA.DIF. Se ora si rilancia il programma e si dispone il file in ordine numerico, il programma genererà il nuovo file NPIOGGIA.DIF.

Si scriva il programma nel computer di cui si dispone, se ne memorizzi una copia su disco e lo si lanci due volte, prima per un ordinamento alfabetico e poi per uno numerico. Quando si è terminato, si guardi il catalogo del disco; ci dovrebbero essere tutti e tre questi file:

PIOGGIA.DIF
APIOGGIA.DIF
NPIOGGIA.DIF

Torniamo ora al VisiCalc per dare uno sguardo ai nuovi file creati. Cominceremo con APIOGGIA.DIF. Si controlli che nel drive ci sia il disco giusto e si battano i seguenti comandi del VisiCalc:

/S#	{richiama il comando di memorizzazione DIF}
L	{per **caricare** un file DIF}
APIOGGIA.DIF (return)	{nome del file}
R	{per caricare il file per **righe**}
/GFI	{formato globale intero}

I risultati compaiono nella Figura 8.3. Come si può vedere, le righe dei dati sono effettivamente state disposte nell'ordine alfabetico dei nomi delle città. Ora cancelliamo lo schermo ed esaminiamo il file in ordine numerico:

/CY	{cancella il foglio}
/S#	{richiama il comando di memorizzazione DIF}
L	{per **caricare** un file DIF}
NPIOGGIA.DIF (return)	{nome del file}
R	{per caricare per **righe**}
/GFI	{formato globale intero}

Si faccia scorrere la finestra verso destra finchè non compare la colonna N. Si vedrà che le righe dei dati sono state ordinate in base ai totali di questa colonna. L'intero foglio compare nella Figura 8.4.

Abbiamo così visto che tutto il processo di ordinamento, da quando si è creato il file DIF originale dal foglio del VisiCalc, a quando si è ricaricato il file DIF ordinato nello schermo, richiede solo un paio di minuti. Il procedimento è molto semplice e i risultati sono piuttosto spettacolari. Ancor più importante è che questo procedimento funziona per *qualsiasi* foglio, con qualunque numero di righe o

	A	B	C	D	E	F	
1	CLEVELND	7	6	8	9	9	8
2	DENVER	2	2	3	5	7	5
3	HONOLULU	11	6	8	4	3	1
4	LOS ANGL	8	7	6	3	0	0
5	MIAMI	6	5	5	9	15	23
6	NEW ORLN	11	12	14	11	11	12
7	NEW YORK	7	7	9	8	9	8
8	SAN FRAN	11	8	6	4	1	0
9	ST.LOUIS	5	5	8	10	10	11
10	WASH DC	7	6	8	7	10	9

	H	I	J	K	L	M	N
1	4	8	7	7	7	6	85
2	5	3	3	3	2	1	39
3	2	2	2	4	8	9	59
4	0	0	1	1	5	6	36
5	18	17	22	21	7	4	152
6	17	13	14	6	10	13	144
7	9	10	8	7	10	9	102
8	0	0	1	3	6	10	50
9	9	7	7	7	6	5	91
10	10	12	8	7	7	8	99

Figura 8.3 - Il foglio delle precipitazioni in ordine alfabetico

	A	B	C	D	E	F	G
1	MIAMI	6	5	5	9	15	23
2	NEW ORLN	11	12	14	11	11	12
3	NEW YORK	7	7	9	8	9	8
4	WASH DC	7	6	8	7	10	9
5	ST.LOUIS	5	5	8	10	10	11
6	CLEVELND	7	6	8	9	9	8
7	HONOLULU	11	6	8	4	3	1
8	SAN FRAN	11	8	6	4	1	0
9	DENVER	2	2	3	5	7	5
10	LOS ANGL	8	7	6	3	0	0

	H	I	J	K	L	M	N
1	18	17	22	21	7	4	152
2	17	13	14	6	10	13	144
3	9	10	8	7	10	9	102
4	10	12	8	7	7	8	99
5	9	7	7	7	6	5	91
6	4	8	7	7	7	6	85
7	2	2	2	4	8	9	59
8	0	0	1	3	6	10	50
9	5	3	3	3	2	1	39
10	0	0	1	1	5	6	36

Figura 8.4 - Il foglio delle precipitazioni in ordine numerico

colonne, purchè contenga le due colonne di dati necessarie, una prima colonna di stringhe e un'ultima colonna di totali.

Nelle rimanenti sezioni del capitolo analizzeremo il programma in BASIC che effettua l'ordinamento. Poichè questo è un po' più lungo dei due visti in precedenza, cominceremo dando un'occhiata alla sua *struttura* prima di esaminarne dettagliatamente le varie subroutine.

LA STRUTTURA DEL PROGRAMMA

Il diagramma della Figura 8.5 è una *tavola strutturale* del programma di ordinamento di file DIF. Il suo scopo è di mostrare graficamente l'organizzazione del programma.

Ciascun riquadro della tavola strutturale rappresenta una delle subroutine del programma e le linee che uniscono i riquadri rappresentano i richiami delle subroutine. Per esempio, si può vedere che la subroutine chiamata "lettura stringhe" richiama la subroutine "lettura elemento dati". Le linee tratteggiate dividono le subroutine a seconda delle tre funzioni principali del programma: lettura e memorizzazione dei dati DIF; loro ordinamento; scrittura del nuovo file. La tavola strutturale, quindi, è uno strumento che consente di vedere come le varie subroutine funzionino tutte insieme. Converrà consultarla spesso, quando analizzeremo l'organizzazione del programma.

La tavola strutturale dà anche una chiara immagine di come la sezione del "programma principale" controlli l'azione del programma. Il programma principale è costituito dai righi da 10 a 250: la prima metà di questa sezione (da 10 a 130) è molto simile al programma principale del programma delle statistiche del Capitolo 7. Questa parte effettua cinque operazioni: stampa un titolo; chiede che venga inserito il nome di un file DIF; apre il file; legge i dati inserendoli nelle matrici T1 e T1$ e, finalmente, chiude il file. Le tre subroutine che controllano la lettura del file DIF (ai righi 300, 400 e 500) sono identiche a quelle del programma delle statistiche.

Dal rigo 140 in poi questo programma di ordinamento assume una personalità propria. Il rigo 140 contiene l'istruzione di INPUT che domanda il tipo di ordinamento desiderato:

```
140 INPUT "A(LFA O N(NUMERICO? ";A$
```

Alla variabile A$ viene inserita "A" o "N". La logica della subroutine di ordinamento dipende da questa lettera. Se viene accidentalmente battuto un carattere diverso, il programma deve avere un modo per informare dell'errore e far ripetere l'operazione di input. La responsabilità di questo controllo è del rigo 150:

```
150 IF A$ <> "A" AND A$ <> "N" THEN 140
```

Se il carattere inserito per A$ non è nè A nè N, il rigo 150 restituisce il controllo del

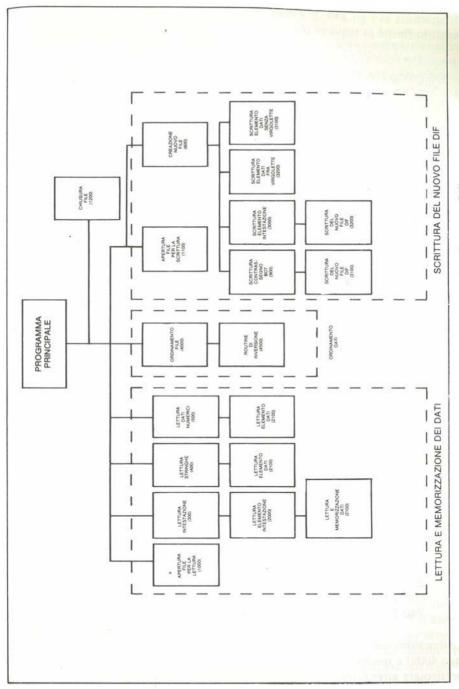



Figura 8.5 - Tavola strutturale del programma di ordinamento DIF

programma al rigo 140. Il risultato di questi due righi è che il messaggio viene ripetuto finchè la risposta fornita non è valida. Per esempio:

```
ORDINAMENTO A(LFA O N(UMERICO? G
ORDINAMENTO A(LFA O N(UMERICO? M
ORDINAMENTO A(LFA O N(UMERICO? A
```

Una volta che il programma conosce il tipo di ordinamento desiderato, è pronto a entrare in funzione. Prima stabilisce il nome del nuovo file DIF da creare, aggiungendo una "A" o una "N" all'inizio del nome originale. Questo avviene nel rigo 160:

```
160 F$ = A$ + F$
```

È un esempio di *concatenazione*, l'unione di due stringhe per formarne una terza. Poichè A$ contiene il codice di ordinamento, "A" o "N", la nuova F$ conterrà il nome corretto del nuovo file.

I righi successivi richiamano le subroutine appropriate e generano sullo schermo messaggi che comunicano cosa sta facendo il programma. Il rigo 180 richiama la subroutine di ordinamento:

```
170 PRINT:PRINT "FILE IN ORDINAMENTO"
180 GOSUB 4000
```

Il nuovo file viene rispettivamente aperto e scritto dalle subroutine richiamate ai righi 200 e 210:

```
190 PRINT:PRINT "FILE IN SCRITTURA"
200 GOSUB 1100
210 GOSUB 600
```

E finalmente, dopo che il rigo 220 ha chiuso il file, i righi 230 e 240 comunicano che il nuovo file ordinato è stato creato:

```
220 GOSUB 1200
230 PRINT:PRINT "*******************"
240 PRINT:PRINT "FILE";F$;"MEMORIZZATO"
```

Le due subroutine introdotte in questo programma sono quella di ordinamento (al rigo 4000) e quella che crea (cioè scrive) il file (al rigo 600). Le esamineremo con particolare attenzione.

LA SUBROUTINE DI ORDINAMENTO

I metodi (o "algoritmi", nella terminologia dei computer) per ordinare un elenco di dati sono molti. Quello utilizzato in questo programma non è il più efficiente in termini di tempo di esecuzione, ma è di gran lunga il più semplice. Viene chiamato "ordinamento a bolla". La sua tecnica consiste nel mettere a confronto ogni elemento dell'elenco, dall'inizio alla fine, con ogni elemento sottostante. Se due elementi non sono nell'ordine desiderato, vengono scambiati di posizione. Quando il procedimento arriva alla fine dell'elenco, l'ordinamento è completo.

I confronti sono controllati da una coppia di loop FOR nidificati:

```
4030    FOR I = 1 TO V-1
4040    FOR J = I+1 TO V
        ...
        (confronto degli elementi ed eventuale inversione)
        ...
4110    NEXT J
4120    NEXT I
```

Si può vedere come questi loop FOR coprano tutti i V vettori dei dati. Poichè i nostri dati sono memorizzati in matrici, le variabili I e J (che sono incrementate dai loop FOR) diventano gli indici delle matrici.

Quello che rende più complessa la routine di ordinamento è che deve essere in grado di disporre i dati sia in ordine alfabetico che numerico, a seconda del valore della variabile del codice di ordinamento, A$. Per questo le istruzioni all'interno dei due loop sono divise in due parti logiche. Il rigo 4050 stabilisce qual è l'insieme di istruzioni da eseguire:

```
4050 IF A$ = "A" THEN GOTO 4090
```

L'ordinamento alfabetico richiede confronti sulla matrice di stringhe, T1$. Se due elementi di questa matrice non sono in ordine, vanno invertite *tutte le righe* che rappresentano. Per rendere il procedimento più semplice, l'inversione viene effettuata da una subroutine diversa, al rigo 4500. Il rigo 4100 la richiama quando il confronto mostra che due elementi di T1$ sono in ordine sbagliato:

```
4100 IF T1$(I)>T1$(J) THEN GOSUB 4500
```

Analogamente, l'ordinamento numerico deve richiamare la subroutine di inversione se due elementi dell'*ultima* colonna di dati sono in ordine sbagliato. Per accedere a quest'ultima colonna viene utilizzato il valore della variabile T, il numero degli esimi *numerici*. (Si ricorderà che la subroutine del rigo 300 stabilisce questo valore dall'elemento TUPLES dell'intestazione). I confronti per l'ordinamento numerico vengono effettuati al rigo 4070:

```
4070 IF T1(T,I)<T1(T,J) THEN GOSUB 4500
```

Se viene effettuato un ordinamento numerico, il programma deve saltare le istruzioni che effettuano quello alfabetico: questo è il motivo dell'istruzione GOTO del rigo 4080:

```
4080 GOTO 4110
```

Se si hanno difficoltà a capire subito la logica dell'ordinamento a bolla, si provi a considerare le matrici T1 e T1 $ come un'unica tavola bidimensionale, come quella del foglio elettronico. Si pensi di confrontare gli elementi lungo una colonna, quella delle stringhe o quella dei totali, e di effettuare le necessarie inversioni. Va però ricordato che quando due elementi vengono scambiati, non basta effettuare l'inversione della colonna di cui si mettono a confronto gli elementi, ma si devono scambiare *le intere righe* rappresentate dai due elementi. Questo è compito della routine di inversione del rigo 4500.
La routine di inversione ha due sezioni. La prima, dal rigo 4510 al 4530, inverte le stringhe della matrice monodimensionale T1$. Il procedimento di inversione è in tre passaggi: primo, uno dei due elementi viene temporaneamente assegnato a una variabile di "parcheggio", H$:

```
4510 H$ = T1$(I)
```

Poi alla posizone del primo elemento può essere assegnato un nuovo valore, quello del secondo:

```
4520 T1$(I) = T1$(J)
```

Finalmente il valore memorizzato in H$ diventa il nuovo secondo elemento:

```
4530 T1$(J) = H$
```

La seconda sezione della routine di inversione effettua l'inversione numerica, anche questa divisa in tre passaggi. Poichè, però, vanno invertite due intere righe di elementi, l'inversione va effettuata all'interno di un loop FOR:

```
4540 FOR K = 1 TO T
4550 H = T1(K,I)
4560 T1(K,I) = T1(K,J)
4570 T1(K,J) = H
4580 NEXT K
```

In termini di tempo di programmazione, l'ordinamento a bolla è molto economico. Anche con l'ulteriore complessità data dalla scelta fra ordinamento numerico e alfabetico, tutta l'operazione richiede meno di venti righi di programma. Ha però i suoi svantaggi: diventa sempre più lento all'aumentare delle dimensioni del file. Se si deve usare il programma di ordinamento per fogli elettronici con molti dati, può convenire considerare la possibilità di utilizzare altri algoritmi. Per modificare il metodo di ordinamento, basta modificare la subroutine di ordinamento; quella di inversione rimarrà uguale.

LA SUBROUTINE PER LA CREAZIONE DEL FILE DIF

Dopo che la matrice è stata ordinata, la subroutine del rigo 600 assume il controllo per scrivere i dati ordinati in un nuovo file DIF. Con un'importante eccezione, il procedimento di scrittura è semplicemente l'inverso di quello di lettura. Per leggere il file DIF abbiamo iniziato richiamando le subroutine di lettura, poi abbiamo inserito i dati DIF in variabili. Adesso, per scrivere un file DIF, cominceremo assegnando valori alle variabili appropriate, *poi* richiameremo la subroutine di scrittura, che scriverà i valori di quelle variabili nel file.

La differenza importantissima fra i due procedimenti è la seguente: nel leggere i file DIF si poteva considerare spazzatura tutti i dati che non ci interessavano, mentre per scrivere correttamente un file DIF è necessario seguire sistematicamente tutti gli elementi del formato DIF e inserirli nel file uno per uno.

Si ricorderà che la nostra biblioteca delle subroutine per la gestione dei file contiene tre routine di scrittura. Quella del rigo 3000 scrive un elemento dell'intestazione; quelle dei righi 3100 e 3200 scrivono entrambe elementi di dati, la prima senza virgolette e la seconda con. Le variabili utilizzate da queste subroutine sono quelle utilizzate dalle subroutine di lettura: T$ per i titoli dell'intestazione, V1 e V2 per i dati numerici, S$ per i dati a stringa. Quindi è a queste quattro variabili che dovremo assegnare valori, prima di ciascun richiamo di una subroutine di scrittura.

La subroutine del rigo 600 è suddivisa in tre parti: la prima (dal rigo 610 al 660) scrive la sezione di intestazione. La seconda (da 670 a 720) e la terza (da 730 a 790) scrivono la sezione dati, prima l'esimo di stringhe e poi quelli numerici. Esaminiamo una parte per volta.

I valori di V1 e S$ sono costanti per tutti gli elementi della sezione di intestazione. Il rigo 620 assegna questi valori:

```
620 V1 = 0 : S$ = " "
```

Poi i quattro righi successivi scrivono ognuno dei quattro elementi della sezione di intestazione. Ciascun rigo assegna nuovi valori a T$ e V2, poi richiama la subroutine in 3000 per scrivere l'elemento. Si noti che i valori di V e T+1 forniscono

rispettivaemnte il numero dei vettori e degli esimi:

```
640 T$ = "VECTORS":V2 = V:GOSUB 3000
650 T$ = "TUPLES":V2 = T+1:GOSUB 3000
```

Ognuno degli esimi della sezione dati del file deve iniziare con un contrassegno BOT. La subroutine in 900 ha lo scopo di scrivere quel contrassegno: assegna i valori appropriati e richiama la subroutine in 3100 per scrivere l'elemento:

```
910 V1 = —1:V2 = 0:S$ = "BOT": GOSUB 3100
```

La seconda parte della subroutine per la creazione del file scrive l'esimo della stringa. Inizia con un richiamo della subroutine del contrassegno BOT:

```
680 GOSUB 900
```

poi assegna valori a V1 e V2. Si ricordi che l'identificatore di tipo per elementi a stringa è 1:

```
690 V1 = 1:V2 = 0
```

Infine, un loop FOR assegna ciascun valore della matrice T1$ e poi scrive l'elemento:

```
700 FOR I = 1 TO V
710 S$ = T1$(I):GOSUB 3200
720 NEXT I
```

Si noti che la subroutine in 3200 viene utilizzata per scrivere gli elementi a stringa. Questa subroutine mette il valore di S$ fra virgolette.

La terza parte di questa subroutine scrive gli esimi numerici tramite una coppia di loop nidificati. Il loop esterno crea ciascun esimo, cominciando con un contrassegno BOT:

```
740 FOR I = 1 TO T
750 GOSUB 900
```

L'identificatore di tipo per i dati numerici è 0; questo valore viene assegnato a V1 e "V" viene assegnato alla variabile a stringa S$:

```
755 V1 = 0:S$ = "V"
```

Il loop interno assegna valori della matrice T1 alla variabile V2 e scrive ciascun

elemento di dati nel file:

```
760 FOR J = 1 TO V
770 V2 = T1(I,J):GOSUB 3100
780 NEXT J
```

C'è ancora un dettaglio, prima che il file possa dirsi completo. Gli ultimi due righi della subroutine scrivono il contrassegno EOD:

```
800 V1 = -1: V2 = 0:S$ = "EOD"
810 GOSUB 3100
```

Tutto sommato, questa subroutine mostra che la scrittura di un file DIF è solo un po' più complicata che non la sua lettura. Come per tutte le operazioni di programmazione, la chiave sta in un'accurata organizzazione e in un oculato uso delle subroutine.

SOMMARIO

Scopo finale dell'associazione VisiCalc/BASIC è di chiudere il cerchio dell'elaborazione dei dati: dal VisiCalc al BASIC e di nuovo al VisiCalc. Come abbiamo visto nel programma di ordinamento, il file DIF costituisce un mezzo eccellente per spostare i dati su questo cerchio, in modo efficiente e affidabile.

APPENDICE A

SUBROUTINE PER LA GESTIONE DEI FILE SU TRE DEI PERSONAL COMPUTER PIU' DIFFUSI

INTRODUZIONE

Le subroutine di questa appendice hanno lo scopo di soddisfare le necessità di gestione dei file dei programmi in BASIC dei Capitoli 6, 7 e 8. Per una descrizione generale dell'operato di queste routine, vedi Figura 6.7 nel Capitolo 6.

I file sequenziali possono essere aperti per l'input o per l'output. Se si sta scrivendo su un file dal quale si vuole poi leggere, è necessario chiuderlo e riaprirlo per la lettura. I programmi dei Capitoli 6 e 7 leggono soltanto, quindi basta una subroutine di apertura. Il programma del Capitolo 8, invece, legge un file, lo chiude e poi ne apre un altro per scrivervi. Questo programma richiede perciò due diverse subroutine di apertura di file, una per la lettura e una per la scrittura; in BASIC, le istruzioni per effettuare le due operazioni non sono le stesse.

Quando un programma legge un file sequenziale, legge *ciascun* dato, uno dopo l'altro, anche se alcuni dati non interessano. Di conseguenza, i programmi dei Capitoli 6, 7 e 8 considerano molti elementi "spazzatura", cioè li leggono e li scartano.

IL BASIC DELL'APPLE®

La Figura A.1 mostra le subroutine per il BASIC Applesoft. In questa versione del BASIC, il carattere control-D informa il sistema operativo che sta per essere eseguito un comando DOS dall'interno di un programma in BASIC. Il codice ASCII di control-D è 4, generato nelle subroutine da CHR$(4). Quindi i comandi OPEN (apertura), CLOSE (chiusura), READ (lettura), e WRITE (scrittura) vengono impartiti da istruzioni PRINT che iniziano con control-D. Per aprire un file sequenziale sono necessarie due istruzioni PRINT, una per il comando OPEN e una

per il comando READ o WRITE. Per esempio:

```
1010 D$ = CHR$(4)
1020 PRINT D$;"OPEN";F$
1030 PRINT D$;"READ";F$
```

Dopo che il file è stato aperto per la lettura, le successive istruzioni di input leggono i dati *dal file*. Analogamente, quando un file è stato aperto per la scrittura, le istruzioni PRINT scrivono dati *sul file*.

```
1000 REM *** APERTURA FILE PER LA LETTURA
1010 D$ = CHR$(4) : REM ** CONTROL-D
1020 PRINT D$; "APERTURA "; F$
1030 PRINT D$; "LETTURA "; F$
1040 RETURN

1100 REM *** APERTURA FILE PER LA SCRITTURA
1110 D$ = CHR$(4) : REM ** CONTROL-D
1120 PRINT D$; "APERTURA "; F$
1130 PRINT D$; "SCRITTURA "; F$
1140 RETURN

1200 REM *** CHIUSURA FILE
1210 PRINT D$ "CHIUSURA"
1220 RETURN

2000 REM *** LETTURA ELEMENTO INTESTAZIONE
2010 INPUT T$
2020 GOSUB 2100
2030 RETURN

2100 REM *** LETTURA ELEMENTO DATI
2110 INPUT V1, V2
2120 INPUT S$
2130 RETURN

3000 REM *** SCRITTURA ELEMENTO INTESTAZIONE
3010 PRINT T$
3020 GOSUB 3200
3030 RETURN

3100 REM *** SCRITTURA ELEMENTO DATI SENZA VIRGOLETTE
3110 PRINT V1; ","; V2
3120 PRINT S$
3130 RETURN

3200 REM *** SCRITTURA ELEMENTO DATI FRA VIRGOLETTE
3210 PRINT V1; ","; V2
3220 PRINT CHR$ (34); S$; CHRS$ (34)
3230 RETURN
```

Figura A.1 - Subroutine per la gestione dei file in Applesoft BASIC

IL BASIC DEL TRS-80

La Figura A.2 mostra le stesse subroutine per il BASIC del TRS-80. Il comando OPEN richiede tre parametri. Il primo, "I" o "O", indica se il file viene aperto per l'input o per l'output. Il secondo, un numero intero fra 1 e 15, assegna il file a un *buffer*, che memorizza i dati temporaneamente, mentre entrano nel file su disco o ne escono. Il terzo parametro è il nome del file. Quindi il comando:

```
1010 OPEN "I",1,F$
```

apre il file F$ per l'input (cioè per la lettura) e lo assegna al buffer n. #1.

```
1000 REM *** APERTURA FILE PER LETTURA
1010 OPEN "I", 1, F$
1020 RETURN

1100 REM *** APERTURA FILE PER SCRITTURA
1110 OPEN "O", 2, F$
1120 RETURN

1200 REM *** CHIUSURA FILE
1210 CLOSE
1220 RETURN

2000 REM *** LETTURA ELEMENTO INTESTAZIONE
2010 INPUT #1, T$
2020 GOSUB 2100
2030 RETURN

2100 REM *** LETTURA ELEMENTO DATI
2110 INPUT #1, V1, V2
2120 INPUT #1, S$
2130 RETURN

3000 REM *** SCRITTURA ELEMENTO INTESTAZIONE
3010 PRINT #2, T$
3020 GOSUB 3200
3030 RETURN

3100 REM *** SCRITTURA ELEMENTO DATI SENZA VIRGOLETTE
3110 PRINT #2, V1; ","; V2
3120 PRINT #2, S$
3130 RETURN

3200 REM *** SCRITTURA ELEMENTO DATI TRA VIRGOLETTE
3210 PRINT #2, V1; ","; V2
3220 PRINT #2, CHR$(34); S$; CHR$(34)
3230 RETURN
```

Figura A.2 - Subroutine per la gestione dei file per il TRS-80

L'istruzione:

```
1110 OPEN "O",2,F$
```

apre il file F$ per l'output e lo assegna al buffer n. #2.

Le istruzioni INPUT# e PRINT# vengono utilizzate per l'input e l'output relativo al file e devono contenere il numero di buffer del file che viene letto o sul quale si scrive. Per esempio:

```
2010 INPUT#1, T$
```

Legge un valore a stringa dal file assegnato al buffer n. #1.

```
1000 REM *** APERTURA FILE PER LETTURA
1010 OPEN F$ FOR INPUT AS #1
1020 RETURN

1100 REM *** APERTURA FILE PER SCRITTURA
1110 OPEN F$ FOR OUTPUT AS #2
1120 RETURN

1200 REM *** CHIUSURA FILE
1210 CLOSE
1220 RETURN

2000 REM *** LETTURA ELEMENTO INTESTAZIONE
2010 INPUT #1, T$
20 GOSUB 2100
2030 RETURN

2100 REM *** LETTURA ELEMENTO DATI
2110 INPUT #1, V1, V2
2120 INPUT #1, S$
2130 RETURN

3000 REM *** SCRITTURA ELEMENTO INTESTAZIONE
3010 PRINT #2, T$
3020 GOSUB 3200
3030 RETURN

3100 REM *** SCRITTURA ELEMENTO DATI SENZA VIRGOLETTE
3110 PRINT #2, V1; ","; V2
3120 PRINT #2, S$
3130 RETURN

3200 REM *** SCRITTURA ELEMENTO DATI TRA VIRGOLETTE
3210 PRINT #2, V1; ","; V2
3220 PRINT #2, CHR$(34); S$;CHR$(34)
3230 RETURN
```

Figura A.3 - Scrittura per la gestione dei file per il Personal Computer IBM

IL BASIC DEL PERSONAL COMPUTER IBM®

Le subroutine relative ai file per il Personal Computer IBM, che compaiono nella Figura A.3, sono simili a quelle del TRS-80. Il Personal Computer dell'IBM dispone di due versioni dell'istruzione di apertura. Per esempio, entrambe le istruzioni che seguono aprono il file F$ per l'input e lo assegnano al buffer n. #1:

```
1010 OPEN F$ FOR INPUT AS #1
1010 OPEN "I",#1,F$
```

Le subroutine della Figura A.3 utilizzano la prima versione, poichè il suo significato è più chiaro di quello della seconda.

APPENDICE B

UNO SGUARDO AL BASIC

INTRODUZIONE

Questa appendice intende aiutare a capire i tre programmi in BASIC presentati nel libro: non vuol essere un'introduzione esauriente al linguaggio BASIC, ma un inizio per chi ha poca o nessuna esperienza della programmazione in BASIC. Il momento migliore per leggere questa appendice è dopo aver completato i primi cinque capitoli prima di cominciare gli ultimi tre.

Le possibilità del BASIC verranno spesso messe a confronto con quelle del VisiCalc: noteremo molte somiglianze fra i due strumenti, ma anche alcune importanti differenze.

I punti importanti di confronto fra il VisiCalc e il BASIC vengono riassunti in neretto.

Due sono i motivi per paragonare le possibilità del VisiCalc a quelle del BASIC:

1. Probabilmente si scoprirà che la conoscenza acquisita del VisiCalc aiuta a "visualizzare" alcuni dei concetti della programmazione in BASIC, rendendo così più semplice l'apprendimento di questo linguaggio.

2. Per poter usare nel modo più efficiente sia il VisiCalc che il BASIC, è necessario saperne abbastanza delle rispettive caratteristiche così da riconoscere quando conviene usare uno o l'altro.

Questo secondo aspetto è uno dei temi dei Capitoli 6, 7 e 8, ma questa appendice fornirà alcuni dettagli tecnici impliciti nella discussione.

COS'E' UN PROGRAMMA IN BASIC

Un programma in BASIC è una sequenza numerata di istruzioni che ha lo scopo di far svolgere al computer un determinato compito. Le istruzioni devono essere tutte scritte nel vocabolario e nella sintassi, limitati e specifici, che costituiscono il

linguaggio BASIC . I righi di un programma sono numerati in ordine ascendente, ma la numerazione non deve necessariamente procedere secondo multipli regolari. Per esempio, i primi righi di un programma potrebbero essere numerati:

```
1
2
3
4
5
```

o:

```
10
20
30
40
50
```

o addirittura:

```
10
13
20
25
37
```

La sola cosa che conta è che i numeri siano in ordine *crescente*, dall'inizio alla fine del programma. Un programma può avere qualsiasi lunghezza sia necessaria per svolgere il compito desiderato, da un solo rigo a diverse centinaia.

Le istruzioni numerate di un programma in BASIC non danno risultati immediati al momento in cui vengono inserite nel computer. Sotto questo aspetto, un programma in BASIC è sostanzialmente diverso da una sequenza di comandi del VisiCalc. In quest'ultimo si ottengono risultati immediati quasi tutte le volte che sul foglio viene inserito un comando: si vedranno nuovi dati, nuovi calcoli o nuovi formati, ma sullo schermo succederà qualcosa di visibile. Nel BASIC, invece, quando si vuole lanciare un programma è necessario impartire al sistema operativo del computer un comando specifico. Di solito per far questo basta battere la parola:

```
RUN
```

quando il programma risiede nella memoria attiva del computer. A questo punto il computer esegue tutte le istruzioni del programma, una alla volta, nell'ordine specificato dalla logica del programma.

Una differenza essenziale fra il BASIC e il VisiCalc è, quindi, che i comandi del VisiCalc vengono generalmente eseguiti non appena sono stati impartiti, mentre quelli del BASIC non vengono eseguiti finchè non viene lanciato l'intero programma.

Un rigo di un programma in BASIC può contenere più di un'istruzione. Le diverse istruzioni di uno stesso rigo sono separate da due punti (:). Per esempio, il rigo che segue contiene tre istruzioni diverse:

```
630 T$ = "TAVOLA":V2=1:GOSUB 3000
```

In alcune versioni del BASIC, le istruzioni su un rigo sono separate dal carattere (\) invece che dai due punti.

Un elenco di tutti i righi del programma si chiama *listato* (*listing*), che può apparire sullo schermo o essere scritto su carta. Spesso nel listato si vorrà inserire una certa quantità di informazioni relative al programma: può darsi che si voglia indicare dove termina una sezione o dove ne inizia un'altra. Oppure si può voler descrivere esattamente qual è la funzione di un determinato rigo. L'istruzione REM (da *remark*, commento) consente di far questo. Quando la prima parola di un rigo è REM, il BASIC ignora tutto il successivo contenuto del rigo. Per esempio, all'inizio di un programma, si può voler scrivere righi che ne descrivono lo scopo:

```
10 REM *** QUESTO PROGRAMMA
20 REM *** MOSTRA SULLO SCHERMO
30 REM *** UN FILE DIF
```

Non ci sono limiti alle parole o ai simboli che si possono scrivere dopo REM. Ci si può mettere tutto quello che si ritiene possa essere utile per chi cerchi di utilizzare o capire il programma. Si può anche usare REM come ultima istruzione di un rigo che contenga più di un'istruzione. Per esempio:

```
70 GOSUB 1000:REM ** APERTURA DEL FILE
```

L'istruzione finale di un programma è END. END comunica al computer che il programma è finito. (Tuttavia non è necessario che END sia nell'ultimo *rigo* di un programma in BASIC. Quando studieremo il modo in cui viene controllata l'esecuzione di un programma, vedremo perchè.)

Finalmente, dopo aver scritto un programma in BASIC, è probabile che lo si voglia memorizzare su disco per poterlo riutilizzare. I programmi possono essere memorizzati su disco e poi "caricati" nuovamente nella memoria attiva del computer, proprio come le istruzioni del VisiCalc. Ciascun sistema operativo ha i suoi comandi per creare su disco un file contenente un programma in BASIC, ed è bene conoscerli *prima* di cominciare a scrivere un programma che meriti di essere memorizzato.

MEMORIZZAZIONE DEI DATI NEL BASIC

Nel VisiCalc, ogni posizione del foglio può contenere un elemento dei dati; anche il BASIC dispone di strutture, chiamate *variabili*, per la memorizzazione dei singoli elementi dei dati. Il motivo di questa denominazione è che i valori memorizzati in una variabile possono cambiare durante l'esecuzione di un programma. Ogni variabile ha un nome, che rimane sempre lo stesso tutte le volte che la variabile viene utilizzata. Molte versioni del BASIC limitano la lunghezza dei nomi delle variabili a uno, due o tre caratteri, il primo dei quali dev'essere una lettera, da A a Z. Esempi di nomi di variabili in BASIC sono:

```
V1
V2
I
J
N
```

Altri BASIC permettono di usare nomi di variabile più lunghi. Il vantaggio che offrono è di poter scrivere nomi che esprimono la natura dei valori contenuti nelle variabili. Per esempio:

```
VALORE1
VALORE2
```

I programmi di questo libro usano nomi di variabile brevi, perchè possano essere accettati dalle versioni di BASIC più comuni. Se si dispone di un BASIC che consente nomi di variabile lunghi, conviene sfruttare questa possibilità, poichè nomi significativi rendono più facile la lettura e la comprensione di un programma.

A differenza delle posizioni di un foglio del VisiCalc, una variabile del BASIC viene definita così da contenere un *tipo* di dati specifico. Questo tipo viene identificato dall'ultimo carattere del nome della variabile: se è $ la variabile serve per la memorizzazione delle stringhe. (I dati non numerici, chiamati *etichette* nel VisiCalc, nel BASIC vengono chiamati stringhe.) Esempi di variabili a stringa sono:

```
T$
S$
A$
```

Un nome che termina col carattere % designa una variabile intera:

```
I%
J%
```

Infine, se un nome di variabile termina con una lettera o una cifra, e non con uno dei

due caratteri speciali ($ o %), indica una variabile *reale*. (Un numero reale può contenere una parte decimale.)

I nomi di variabile del BASIC non vanno confusi con gli indirizzi del VisiCalc. In quest'ultimo, quando si fa riferimento a un indirizzo, come M5, si indica una posizione del foglio, e cioè la posizione all'intersezione della colonna M con la riga 5. Anche una variabile del BASIC potrebbe chiamarsi M5, ma i caratteri di questo nome non avrebbero un significato particolare per il BASIC. Di solito vengono scelti nomi di variabile che aiutano a ricordare quello che le variabili rappresentano; per esempio, M5 può significare "il mese n. 5". Per il BASIC, comunque, M5 è un nome come un altro.

Una variabile del BASIC, come una posizione del foglio del VisiCalc, memorizza un solo dato. Il tipo di dato che la variabile può immagazzinare viene indicato dall'ultimo carattere del nome della variabile. Per il resto, i nomi delle variabili vengono scelti per il significato che hanno per il programmatore, non per un significato per il BASIC.

L'istruzione in BASIC per memorizzare un dato in una variabile si chiama *istruzione di assegnazione*, perchè assegna un valore alla variabile. Un esempio di tale istruzione è:

```
10 V1 = 1
```

che significa: "memorizza il valore 1 nella variabile V1." Il "verbo" di un'istruzione di assegnazione è sempre rappresentato dal segno = alla sinistra di tale segno può esserci solo un unico nome di variabile, mentre alla destra sono legali espressioni più complesse. Per esempio:

```
205 V2 = T+1
```

Questa istruzione coinvolge due variabili, V2 e T. L'istruzione dice: "Addiziona 1 al valore di T e memorizza la somma in V2". Quando l'istruzione viene eseguita, solo la variabile V2 riceve un nuovo valore, mentre la variabile T, alla destra del segno uguale, rimane invariata.

Spesso, in un programma in BASIC, il nome di una variabile compare su entrambi i lati del segno uguale, però in un modo speciale. Il calcolo del nuovo valore sarà in parte basato sul vecchio valore della variabile. Ecco un esempio:

```
215 V1 = V1 + 1
```

Questa istruzione dice: "Addiziona 1 al valore attuale di V1, poi memorizza la somma in V1". Oppure, più semplicemente: "Incrementa di 1 il valore di V1". In questo caso, poichè V1 può memorizzare un solo valore alla volta, il vecchio valore di V1 va perduto.

Non ci sono limiti alla complessità aritmetica della parte destra di un'istruzione di assegnazione. Il BASIC calcola il valore dell'espressione a destra e assegna il risultato alla variabile a sinistra. I simboli utilizzati per le operazioni aritmetiche sono gli stessi del VisiCalc:

+ addizione
— sottrazione
* moltiplicazione
/ divisione
^ elevazione a potenza

L'*ordine in cui tali operazioni vengono calcolate non* è, invece, lo stesso del VisiCalc. Il VisiCalc, come si ricorderà, calcola semplicemente le espressioni aritmetiche da sinistra a destra, mentre il BASIC rispetta una gerarchia di operazioni. Ecco l'ordine di calcolo seguito:

1. elevazione a potenza
2. moltiplicazione e divisione (da sinistra a destra)
3. addizione e sottrazione (da sinistra a destra)

Per esempio, si consideri l'espressione sulla destra della seguente istruzione di assegnazione:

```
100 V1 = A + B * C^D
```

L'espressione, così com'è scritta, verrebbe calcolata, da sinistra a destra, nel seguente ordine: prima l'elevazione a potenza, poi la moltiplicazione e infine l'addizione.

Come nel VisiCalc, è però possibile modificare l'ordine di esecuzione delle operazioni di un'espressione aritmetica tramite parentesi:

```
100 V1 = ((A+B) * C)^D
```

Con queste parentesi l'espressione verrebbe calcolata, da sinistra a destra, a partire dalla parentesi più interna muovendo verso l'esterno.

Il VisiCalc e il BASIC hanno due diversi sistemi di calcolo delle espressioni aritmetiche. Per questo, è necessario scrivere tali espressioni con molta attenzione, per evitare inattesi e spiacevoli risultati.

MATRICI NEL BASIC

Oltre alle variabili semplici, il BASIC dispone di un'altra struttura, più complessa, per la memorizzazione dei dati: la *matrice*. Una matrice immagazzina *gruppi* di dati in modo molto simile alle righe e alle colonne del foglio del VisiCalc.

Per usare una matrice di un programma in BASIC, bisogna innanzi tutto specificarne esplicitamente tre caratteristiche: il tipo, le dimensioni e la lunghezza. A questo scopo, il BASIC ha un'istruzione speciale: DIM (per *dim*ensioni). Ecco un esempio:

```
20 DIM T1$(10), T1(12,10)
```

Questa istruzione DIM definisce due matrici. La prima si chiama T1$ e il nome indica che si tratta di una matrice di stringhe, e cioè che tutti i dati memorizzati in T1$ saranno stringhe. (Le convenzioni per indicare i tipi delle variabili rimangono valide anche per le matrici.) Il numero 10, in parentesi dopo il nome, indica la *lunghezza* della matrice; specificamente, comunica che T1$ è una matrice che può contenere 11 dati, numerati da 0 a 10. T1$ è inoltre una matrice monodimensionale, poichè c'è un solo numero fra parentesi.

La matrice T1 è una matrice bidimensionale di numeri reali. Dato che le lunghezze delle sue due dimensioni vengono rispettivamente indicate in 12 e 10, sappiamo che la matrice può contenere una tavola di numeri reali lunga 13 elementi (da 0 a 12) e larga 11 (da 0 a 10).

Quando si assegnano valori a una matrice, è necessario specificare il punto esatto in cui devono essere memorizzati. Per esempio, le seguenti istruzioni assegnano valori a stringa agli undici elementi della matrice T1$:

```
10 T1$(0) = "ZERO"
15 T1$(1) = "UNO"
20 T1$(2) = "DUE"
25 T1$(3) = "TRE"
30 T1$(4) = "QUATTRO"
35 T1$(5) = "CINQUE"
40 T1$(6) = "SEI"
45 T1$(7) = "SETTE"
50 T1$(8) = "OTTO"
55 T1$(9) = "NOVE".
60 T10$(10) = "DIECI"
```

Si può anche usare una *variabile* per indicare una posizione della matrice. Se, per esempio, si è assegnato, nel rigo 20 del programma, il valore 5 alla variabile I, l'istruzione

```
30 T1$(I) = "CINQUE"
```

assegna una stringa al quinto elemento (I=5) di T1$ e la variabile I di questo esempio viene chiamata l'*indice* della matrice T1$

Una matrice monodimensionale del BASIC è come una riga o una colonna di un foglio elettronico del VisiCalc: memorizza un elenco di dati. Una matrice bidimensionale memorizza una tavola di dati. Ciascuna matrice può però contenere dati di un solo tipo.

INPUT E OUTPUT NEL BASIC

Quando si usa il VisiCalc c'è la tendenza a non considerare l'input e l'output due procedimenti distinti, visto che si svolgono contemporaneamente. Se si crea una tavola di numeri, per esempio, si può veder la tavola che si sviluppa *mentre* si inseriscono i valori se si batte una formula per calcolare un nuovo valore da inserire nella tavola, i risultati della formula compaiono immediatamente.

Nel BASIC, invece, è molto più probabile che il processo di inserimento dei dati nel programma sia chiaramente distinto dal successivo processo di output dei risultati. In effetti, il BASIC ha due insiemi distinti di comandi per l'input e l'output, ed è necessario specificare a un programma come, quando e dove i dati vanno scritti e letti.

Tre sono le fonti possibili di input di dati per un programma in BASIC. Quella più semplice e più statica è il programma stesso. Per esempio, si può usare un'istruzione di assegnazione per rendere alcuni dati disponibili al programma:

```
100 P1 = 3.14
```

Lo svantaggio dello scrivere dati direttamente nel programma è che se poi si vogliono modificare è necessario intervenire sul programma. Se un programma deve ricevere una quantità di dati che saranno probabilmente diversi a ogni esecuzione, questo non è un metodo conveniente.

Un approccio migliore, per il quale il BASIC dei microcomputer è particolarmente indicato, consiste nel dare istruzioni al programma perchè legga i dati dalla tastiera. In effetti, allora si può redigere un programma in BASIC che "parli" con la persona che siede al computer e riceva informazioni da essa tramite la tastiera. Questa comunicazione viene spesso chiamata un *dialogo* fra programma e utente. Tutti gli aspetti di questo dialogo devono essere pianificati in anticipo, come se si scrivesse la sceneggiatura di una commedia.

La principale istruzione per condurre questi dialoghi è INPUT. I particolari del suo funzionamento possono variare leggermente fra le diverse versioni di BASIC, ma le sue funzioni essenziali sono tre:

1. Mostrare un messaggio informativo (*prompt*) sullo schermo. Di solito si tratta di una domanda o di una spiegazione che indica all'utente qual è il dato da

inserire. Questo prompt viene deciso dal programmatore quando scrive l'istruzione INPUT.

2. Leggere il dato che l'utente scrive alla tastiera in risposta al prompt.

3. Assegnare quel dato alla variabile specificata.

Il seguente esempio illustra le tre funzioni:

```
50 INPUT "QUANTI ANNI HAI?"; Y%
```

Questo rigo contiene tre elementi diversi: primo, la parola del comando, INPUT; secondo, il prompt, scritto fra virgolette; terzo, il nome della variabile in cui va memorizzato il dato di input. Quando il BASIC esegue questo rigo, comincia mostrando il prompt sullo schermo:

```
QUANTI ANNI HAI?
```

poi aspetta che venga battuta una risposta alla tastiera. Supponiamo che l'utente batta 31:

```
QUANTI ANNI HAI? 31
```

Il valore fornito, 31, verrà memorizzato nella variabile Y%. Tutte le volte che il programma avrà successivamente bisogno di utilizzare tale dato, lo troverà in Y%.

Naturalmente l'istruzione di input può essere utilizzata per leggere dati a stringa, oltre a quelli numerici. La seguente istruzione inserisce una stringa nella variabile N$:

```
40 INPUT "COME TI CHIAMI?";N$
```

Per l'output di dati sullo schermo, il BASIC fornisce il comando PRINT. Gli usi di PRINT sono molteplici. Anche in questo caso, le diverse versioni del BASIC offrono caratteristiche diverse per il comando PRINT. Il metodo di utilizzazione più semplice consiste nel mostrare sullo schermo il valore corrente di una variabile. L'istruzione:

```
60 PRINT Y%
```

mostra il valore memorizzato nella variabile Y%. In un'istruzione PRINT possono anche essere incluse stringhe, il che torna utile quando si vuol mostrare dati in un contesto esplicativo. Per esempio:

```
60 PRINT "HAI ";Y%;" ANNI"
```

Questa istruzione inserisce il valore di Y% in una frase che appare sullo schermo. Se il valore attualmente memorizzato in Y% è 31, questa istruzione PRINT genera la seguente riga di output:

```
HAI 31 ANNI
```

La punteggiatura di un'istruzione PRINT è sempre significativa. Il punto e virgola (;) serve a separare fra loro gli elementi di un'istruzione PRINT, come nel precedente rigo 60. La virgola svolge un'ulteriore funzione: quando viene usata come separatore in un comando PRINT, genera colonne di dati. Per esempio, supponiamo che le variabili V1, V2, V3, V4, V5 e V6 contengano rispettivamente i valori interi da 1 a 6. Le istruzioni:

```
70 PRINT V1,V2,V3
80 PRINT V4,V5,V6
```

generano le seguenti righe:

```
1    2    3
4    5    6
```

Gli spazi che separano le colonne e il numero di colonne sullo schermo dipendono dalla versione del BASIC utilizzata.

Molti BASIC dispongono di una funzione chiamata TAB che, anche lei, mostra i risultati in formato tabulare. La funzione TAB impone al BASIC di scrivere i dati di output a partire da certe righe dello schermo. Per esempio, i righi:

```
90 PRINT TAB(10); V1; TAB(20); V2
100 PRINT TAB(10); V3; TAB(20); V4
```

generano le due seguenti colonne di dati:

```
1    2
3    4
```

La prima colonna è a 10 spazi dal bordo sinistro dello schermo e le due colonne sono distanziate di 10 spazi.

Infine il comando PRINT USING (stampa usando) offre, per le versioni del BASIC che ne dispongono, un mezzo per determinare il formato dell'output dei dati numerici. PRINT USING prende le istruzioni da una stringa costituita da caratteri speciali (compresi "#", "$" "*", "." e ",") per determinare il formato di un un numero di output. La Figura B.1 contiene esempi e una spiegazione di questo comando.

Il BASIC ha alcune possibilità di rappresentazione sullo schermo simili ai comandi di formato globale del VisiCalc. Queste possibilità comprendono la funzione TAB, che è uno strumento per generare colonne di dati, e PRINT USING, per determinare il formato di dati numerici. In generale, comunque, nel BASIC l'input e l'output vanno progettati in modo molto più esplicito che non nel VisiCalc.

La terza fonte di dati per un programma in BASIC è un file di dati esterno, memorizzato su disco o su cassetta. Tendenzialmente, la sintassi dei comandi di input e di output relativi al file varia molto fra una versione e l'altra del BASIC l'Appendice A mostra esempi di questi comandi per tre BASIC.

```
ISTRUZIONI                                       RISULTATI

10 V1 = 1532.14
20 PRINT USING "$$##,###.##"; V1            $1,532.14

-----------------------------------------------------------------

10 V1 = 23456.789
20 S$ = "PAGA ESATTAMENTE**$##,####.##"
30 PRINT USING S$; V1                      PAGA ESATTAMENTE ***$23,456.79

-----------------------------------------------------------------

10 V1 = 1.234
20 V2 = 25.71
30 V3 = 98.7654
40 S$ = "###.#    ###.#    ###.#"
50 PRINT USING S$; V1, V2, V3               1.2     25.7     98.8
```

#	Rappresenta una cifra del numero
.	Indica la posizione del punto decimale
,	Fa sì che vengano stampate virgole ogni tre cifre verso sinistra a partire dal punto decimale; può essere inserito in qualsiasi punto della stringa di formattazione
$	Drive il simbolo del dollaro nella posizione indicata
$$	Inserisce un simbolò di dollaro "mobile"
**	Riempie con asterischi gli spazi fino alla prima cifra
**$	Scrive asterischi prima del simbolo di dollaro "mobile"

Figura B.1 - Esempi e spiegazione di PRINT USING

Per quanto riguarda l'accesso ai dati, i file che li contengono sono di due tipi: ad accesso sequenziale e ad accesso casuale. I dati dei primi devono essere letti dall'inizio alla fine, uno per uno. In altri termini, se si vuole accedere al decimo dato di un file sequenziale, bisogna prima leggerne i primi nove, anche se questi sono del tutto irrilevanti rispetto al compito in oggetto. I file ad accesso sequenziale si prestano a compiti di elaborazione che implicano la memorizzazione e l'accesso a molti dati simili.

I dati di un file ad accesso casuale possono essere letti in qualsiasi ordine. Spesso un file di questo tipo viene letto in collegamento con una struttura di indicizzazione che facilita il ricupero di dati specifici.

CONTROLLO DI UN PROGRAMMA IN BASIC

Sebbene i righi di un programma in BASIC siano numerati in ordine ascendente, non è necessariamente questo l'ordine in cui vengono eseguite le istruzioni. Spesso la logica di un programma implica un salto a un rigo in posizione più avanzata o il ritorno a uno già superato, e spesso le istruzioni di un rigo, o di un gruppo di righi, vengono eseguite più volte. Il BASIC offre diversi tipi di comandi che consentono al programmatore di controllare l'ordine di esecuzione dei righi di un programma. Tali comandi sono GOTO (vai a), GOSUB (vai a sub) e FOR/NEXT (per/successivo) e verranno adesso esaminati uno per volta.

IL COMANDO GOTO E L'ISTRUZIONE IF

GOTO attribuisce il controllo del programma a un numero di rigo specifico il salto da effettuare può essere in avanti o indietro. La seguente istruzione, sul rigo 10, impone al programma di saltare al rigo 100:

```
10 GOTO 100
```

Questa istruzione fa eseguire un salto all'indietro:

```
200 GOTO 150
```

Quando il BASIC incontra un'istruzione GOTO, salta immediatamente al rigo specificato ed esegue la istruzione (o le istruzioni) che contiene, dopo di che procede in avanti dal rigo in cui si trova.

Il comando GOTO del BASIC non va confuso con il GO TO (>) del VisiCalc. Non hanno niente in comune. Il GOTO del BASIC controlla l'ordine di esecuzione dei righi di un programma il GO TO del VisiCalc porta semplicemente il cursore in una posizione del foglio.

L'azione del comando GOTO può essere sottoposta a *condizioni* se GOTO viene associato a un'istruzione IF (se). Per esempio:

```
50 IF I > 0 THEN GOTO 10
```

Questa istruzione significa: "Se il valore della variabile I è maggiore di 0, allora riporta il controllo del programma al rigo 10, altrimenti procedi come al solito."

Come il VisiCalc, anche il BASIC ha un vocabolario di istruzioni logiche. Oltre a IF, ci sono AND (e), OR (o) e NOT (non) e gli operatori logici >, <, >=, <=, = e < >, che sono concettualmente analoghi alle equivalenti espressioni del VisiCalc. C'è però una differenza importante fra il VisiCalc e il BASIC:

Nel VisiCalc, una funzione @IF effettua semplicemente una scelta fra due valori diversi e ne memorizza uno in una posizione specificata. Nel BASIC, invece, l'azione che risulta da un'istruzione IF può essere espressa praticamente da tutti i comandi BASIC, compresi PRINT, INPUT, GOTO, GOSUB o un'istruzione di assegnazione.

La Figura B.2 mostra alcuni esempi di istruzioni IF nel BASIC. Conviene cercar di capire quali sono i risultati che ognuno di loro produce.

```
10 V1 = 0
20 V2 = 10000
30 INPUT "SCRIVI UN NUMERO";X
40 IF (X < V1) THEN PRINT "IL NUMERO E' NEGATIVO"
50 IF (X > = V1) AND (X < = V2) THEN PRINT "FRA ";V1; "E"; V2
60 IF (X > V2) PRINT "NUMERO GRANDE"
70 IF (X < V1) OR (X> V2) THEN PRINT "NEGATIVO O MOLTO GRANDE"
80 IF NOT (X = V1) THEN PRINT "NON UGUALE A ";V1
90 IF (X < > V2) THEN PRINT "NON UGUALE A ";V2
100 END
```

Figura B.2 - Esempi dell'istruzione IF in BASIC

IL COMANDO GOSUB

Il comando GOSUB del BASIC attribuisce il controllo del programma a una subroutine e si dice che il comando GOSUB "richiama" una subroutine. Una subroutine è un gruppo di righi che svolge un determinato compito; quando questo è stato effettuato, la subroutine *restituisce* il controllo del programma al rigo che segue quello che l'aveva richiamata. Ecco un programma breve, e piuttosto bizzarro, contenente subroutine:

```
10 REM *** INFORMAZIONI PERSONALI
20 INPUT "QUANTI ANNI HAI? "; Y%
30 IF (Y% >= 17) AND (Y% < 21) THEN GOSUB 100
40 IF Y% < 17 THEN GOSUB 200
50 IF Y% >= 21 THEN GOSUB 300
60 INPUT "COME TI CHIAMI? ";N$
70 GOSUB 1000 : REM ** ELABORAZIONE DEI DATI
80 END

100 REM ** ADOLESCENTI
110 INPUT "L(AVORI O VAI A S(CUOLA < L O S( > ? "; A$
120 IF (A$ < > "S") AND (A$ < > "L") THEN GOTO 110
130 IF A$ = "S" THEN GOSUB 200
140 IF A$ = "W" THEN GOSUB 300
150 RETURN

200 REM ** GIOVANI
210 INPUT "DOVE VAI A SCUOLA?"; S$
220 RETURN

300 REM ** ADULTI
310 INPUT "DOVE LAVORI? "; W$
320 INPUT "CHE LAVORO FAI? "; J$
330 RETURN

1000 REM ** ELABORAZIONE DATI
1010 REM
1020 REM -- QUESTA SUBROUTINE NON E' ANCORA STATA SCRITTA
1030 RETURN
```

Figura B.3 - Programma delle "Informazioni personali"

Questo programma ha il compito di raccogliere informazioni su una persona, per poi elaborarle in un determinato modo. Il programma suddivide le persone in tre categorie di età: giovani con meno di 17 anni, che vanno a scuola; adolescenti da 17 a 20 anni, che vanno a scuola o lavorano; e adulti, di 21 anni o più, che lavorano. (Naturalmente questa semplificazione è solo a scopo illustrativo.)

Il programma ha tre subroutine che leggono le informazioni dalla tastiera: la subroutine degli adolescenti, al rigo 100; quella dei giovani, al rigo 200 e quella degli adulti, al rigo 300. Inoltre, la subroutine al rigo 1000 ha lo scopo di elaborare le informazioni. Ognuna delle prime tre subroutine chiede informazioni su una categoria di età specifica. Durante un lancio del programma vengono richiamate solo una subroutine o due. Guardiamo il funzionamento.

Il rigo 20 legge l'età della persona che risponde al programma. Questa età viene memorizzata nella variabile Y%:

```
20 INPUT "QUANTI ANNI HAI? ";Y%
```

I righi 30, 40 e 50 contengono istruzioni IF che sottopongono a un controllo il

valore di Y%. Il rigo 30 verifica se l'età della persona rientra nel campo dei giovani, nel qual caso richiama la subroutine corrispondente:

```
30 IF (Y%>=17) AND (Y%<21) THEN GOSUB 100
```

Se il valore di Y% non rientra nel campo degli adolescenti, il rigo 30 *non fa eseguire nessuna azione*. Il programma passa poi al rigo 40.

I righi 40 e 50 controllano, rispettivamente, se Y% è nel campo dei giovani o se è in quello degli adulti:

```
40 IF Y%<17 THEN GOSUB 200
50 IF Y% >= 21 THEN GOSUB 300
```

Il punto importante è questo: *solo uno* dei tre righi (30, 40 o 50) comporterà il trasferimento del controllo a una subroutine: la decisione di quale si tratta dipende dal valore di Y%. Qualunque sia, la subroutine richiamata svolge il proprio compito di raccolta delle informazioni e poi restituisce il controllo al rigo successivo all'istruzione GOSUB nel quale è stato effettuato il richiamo.

Per esempio, supponiamo che la persona che risponde al programma abbia 31 anni. Il valore di input 31 viene immagazzinato in Y%. Le condizioni IF dei righi 30 e 40 non comportano richiami di subroutine, quindi il programma passa al rigo 50. La condizione:

```
Y%>=21
```

è soddisfatta, poichè 31 è maggiore di 21. Quindi il rigo 50 attribuisce il controllo del programma alla subroutine del rigo 300. La subroutine relativa agli adulti fa due domande, memorizza le risposte. poi restituisce il controllo alla fonte:

```
330 RETURN
```

Come conseguenza, il rigo che viene eseguito dopo il 330 è il 60.

Guardiamo il dialogo specifico che appare sullo schermo:

```
QUANTI ANNI HAI? 31
DOVE LAVORI? SOC XYZ
CHE LAVORO FAI? PROGRAMMATRICE
COME TI CHIAMI? PAT SMITH
```

La prima delle quattro domande deriva dall'istruzione INPUT del rigo 20; la seconda e la terza vengono poste dalla subroutine degli adulti del rigo 300; l'ultima dal rigo 60.

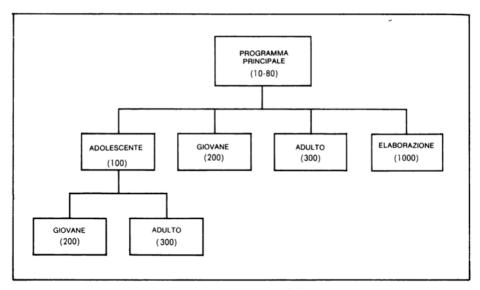



Figura B.3 - Tavola strutturale del programma delle "Informazioni personali"

Guardiamo ora i righi 70 e 80:

```
70 GOSUB 1000
80 END
```

Il rigo 70 richiama un'altra subroutine (quella che elabora i dati) e il rigo 80 chiude il programma. I righi da 10 a 80 costituiscono quello che può essere chiamata la sezione del "programma principale". Sono i righi che controllano, appunto, l'azione principale del programma, richiamando le subroutine. Queste ultime, a loro volta, svolgono compiti specifici per poi restituire il controllo del programma alla sezione del programma principale.

La Figura B.3 mostra un diagramma del modo in cui il programma è organizzato. Tale diagramma viene chiamato *tavola strutturale* ed è uno strumento che permette di visualizzare l'attività del programma. Si può vedere come le quattro subroutine, rappresentate da rettangoli, vengono tutte controllate dalla sezione del programma principale. Si può poi vedere che la subroutine degli adolescenti, del rigo 100, è collegata ad altre due subroutine, perchè richiama quella degli adulti o quella dei giovani per svolgere un determinato compito. Questo illustra un punto importante: le subroutine possono richiamare altre subroutine, come le richiama il programma principale.

Esaminiamo la subroutine degli adolescenti. Inizia, al rigo 110, facendo la domanda:

```
110 INPUT "L)AVORI O VAI A S)CUOLA < L O S>?"; A$
```

Sullo schermo, questa domanda compare come

```
L)AVORI O VAI A S)CUOLA <L O S>?
```

La persona deve rispondere con una L o una S. Il rigo 110 memorizza il carattere di risposta nella variabile A$ al rigo 120 esamina il valore di A$ in un'istruzione IF: se A$ non contiene nè "L" nè "S", il rigo 120 rimanda il controllo del programma al rigo 110 per porre nuovamente la domanda:

```
120 IF (A$< >"S") AND (A$< >"L") THEN GOTO 110
```

A questo punto, a seconda del valore di A$, i righi 130 e 140 richiamano la subroutine dei giovani o quella degli adulti. In altri termini, la subroutine degli adolescenti può portare a uno dei due seguenti dialoghi: se l'adolescente va a scuola, il dialogo viene in parte ripreso dalla subroutine dei giovani:

```
L)AVORI O VAI A S)CUOLA <L O S>? S
DOVE VAI A SCUOLA? SCUOLA ABC
```

Se invece l'adolescente lavora, il dialogo viene dalla subroutine degli adulti:

```
L)AVORI O VAI A S)CUOLA <L O S>? L
DOVE LAVORI? SOC XYZ
CHE LAVORO FAI? PROGRAMMATORE
```

Questo tipo di organizzazione del programma presenta diversi vantaggi. Eccone tre fra i più importanti:

1. Isolando alcuni compiti in subroutine, il programma può essere reso più comprensibile e, più importante, più facilmente modificabile. Supponiamo, per esempio, che si decida di fare altre domande ai giovani. Poichè il programma è organizzato in subroutine, si sa esattamente dove trovare i righi che riguardano i giovani: nella subroutine che inizia al rigo 200. Le modifiche dovrebbero essere apportate solo a questa subroutine, senza riguardare nè la sezione del programma principale nè le altre subroutine del programma.

2. Una subroutine può essere richiamata tutte le volte che si desidera; isolando così compiti ricorrenti in subroutine, si evita di dover riscrivere ogni volta le istruzioni.

3. Quando si sviluppa un programma, si può riservare righi per subroutine che ancora non si sono scritte. Per esempio, guardiamo la subroutine del rigo 1000, quella dell'elaborazione. Questa subroutine contiene solo righi REM,

cioè, quando viene richiamata, non fa nulla; si limita a restituire il controllo al programma principale. Si può avere una miriade di progetti su questa subroutine: forse si vogliono far scrivere le informazioni; o può darsi che le si voglia memorizzare in un file. Qualunque sia la nostra intenzione, ancora non si sono scritte le istruzioni per realizzarle. Il punto importante è invece questo: si può lanciare il programma a questo stadio, verificare le subroutine già scritte e assicurarsi che facciano quello che si vuole. Successivamente, quando si sarà pronti, si può tornare ai righi lasciati per le subroutine di elaborazione (che iniziano da 1000) e scrivere le istruzioni necessarie.

L'ISTRUZIONE FOR/NEXT

Una delle caratteristiche più belle del computer è che esso è disposto a ripetere la stessa operazione centinaia o migliaia di volte senza un lamento. Abbiamo visto come ottenere questo tramite i comandi GOTO o GOSUB.

Quando si conosce esattamente quante volte una determinata sequenza di righi dev'essere ripetuta, il BASIC fornisce le istruzioni FOR/NEXT. La struttura creata da queste istruzioni viene anche chiamata loop (circolo, giro) FOR, perchè il programma torna indietro, in una specie di circolo, per eseguire ripetutamente le stesse istruzioni. Guardiamo un esempio.

Il seguente loop FOR ripete 10 volte un'istruzione PRINT:

```
10 FOR I = 1 TO 10
20 PRINT I
30 NEXT I
```

Questi tre righi faranno apparire sullo schermo la seguente colonna di dieci numeri:

```
1
2
3
4
5
6
7
8
9
10
```

Queste istruzioni FOR e NEXT impartiscono al computer diversi comandi che possono essere suddivisi in quattro passaggi:

1. Assegnazione del valore 1 alla variabile I (chiamata anche l'*indice* del loop FOR).

2. Esecuzione delle istruzioni fra il rigo contenente FOR e quello contenente NEXT. (In questo caso c'è un solo rigo, il 20, con l'istruzione PRINT.)

3. Aumento di 1 del valore di I.

4. Se I è uguale a 10 o minore, ripetizione della sequenza, a partire dal passaggio #2. Se invece I è maggiore di 10, l'azione del loop è terminata.

In questo esempio il valore dell'indice I, viene incrementato di 1 a ogni ripetizione del loop FOR. Il BASIC consente di modificare il valore di questo incremento aggiungendo all'istruzione FOR la precisazione STEP (gradino):

```
10 FOR I = 2 TO 10 STEP 2
20 PRINT I
30 NEXT I
```

L'output conseguente di questi tre righi è:

```
2
4
6
8
10
```

(Se il perchè non è chiaro, si torni ai quattro passaggi descritti precedentemente. Al passaggio 1, a I viene inizialmente assegnato il valore 2; al passaggio 3, I viene aumentato di 2 invece che di 1.)

Il loop FOR del BASIC ha un'altra caratteristica molto importante: il campo degli indici può essere definito tramite variabili, invece che con numeri espliciti. Per esempio:

```
10 FOR I = V1 TO V2 STEP V3
...
50 NEXT I
```

In questo caso l'azione del loop FOR dipende dai valori memorizzati nelle variabili V1, V2 e V3. All'indice, I, viene inizialmente assegnato il valore V1, poi, a ogni ripetizione del loop FOR, I viene incrementato del valore di V3. Il processo ha termine quando I è maggiore di V2.

Queste variabili permettono al loop FOR di essere aperto, cioè, il numero delle ripetizioni può essere stabilito *durante* l'esecuzione del programma, che può leggere

i valori di V1, V2 e V3 dalla tastiera o da un file, oppure può calcolarli da altri dati.

Un loop FOR rappresenta anche un comodo metodo per accedere ai valori di una matrice. Si ripensi alla matrice di stringhe T1$ in cui abbiamo memorizzato i valori a stringa da "ZERO" a "DIECI". A questi valori si può accedere nel seguente modo:

```
100 FOR I = 1 TO 10
110 PRINT T1$(I)
120 NEXT I
```

I risultati ottenuti sarebbero:

```
ZERO
UNO
DUE
TRE
QUATTRO
CINQUE
SEI
SETTE
OTTO
NOVE
DIECI
```

Si noti come la variabile I, che viene incrementata dal loop FOR, venga usata come indice per accedere alla matrice T1 $. I loop FOR e le matrici possono formare una combinazione di strumenti di programmazione estremamente efficace.

Il loop FOR del BASIC ha chiaramente una funzione simile al comando di replica del VisiCalc e; sono tutti e due in grado di applicare ripetutamente una formula a un insieme di dati. Però, in molte occasioni, la caratteristica dell'apertura rende il loop FOR molto più versatile ed efficace del comando /R del VisiCalc.

Fra il rigo FOR e quello NEXT, un loop FOR può contenere istruzioni di qualsiasi tipo, compreso un altro loop FOR. Loop FOR all'interno di altri formano un *nido di loop*:

```
10 FOR I = 1 TO 10
20 FOR J = 1 TO 10
 ...
50 NEXT J
60 NEXT I
```

Il far rientrare le scritte è facoltativo ma aiuta a rendersi conto con un'occhiata del modo in cui i loop sono organizzati.

FUNZIONI NEL BASIC

La maggior parte delle versioni del BASIC dispongono degli stessi tipi di funzioni del VisiCalc e per esempio, sono generalmente comprese le funzioni trigonometriche e quelle esponenziali. Per duplicarne altre, invece, come @SUM, @AVE e @NPV, è necessario scrivere semplici routine che utilizzano i loop FOR e le matrici. Per esempio, se la matrice T1 contiene 10 valori, il seguente loop FOR trova la loro media:

```
100 A = 0
110 FOR I = 1 TO 10
120 A = A + T1(I)
130 NEXT I
140 A = A/10
```

Il rigo 120 accumula la somma dei valori della matrice T1. (Si noti come, nel rigo 100, ad A venga assegnato il valore *iniziale* di 0.) Dopo che il loop FOR ha effettuato l'addizione, il rigo 140 trova la media.

Il BASIC dispone anche di una quantità di funzioni variabili che operano su stringhe piuttosto che su numeri. Ogni versione del BASIC ha il proprio insieme di funzioni a stringa.

Infine, la maggior parte dei BASIC consentono di scrivere funzioni monolineari particolari, chiamate "funzioni definite dall'utente". Le si crea con l'istruzione DEF FN, e i due seguenti righi mostrano esempi sia della creazione che del successivo uso di una funzione del genere:

```
10 DEF FNS(X) = (X+1)^2
20 PRINT FNS(5)
```

Il rigo 10 definisce la funzione FNS. Questa funzione riceve un valore nella variabile "fittizia" X, aggiunge 1 al valore ed eleva il risultato al quadrato. Il rigo 20 fornisce alla funzione il valore 5 e stampa il valore calcolato dalla funzione. Il risultato sarà:

```
36
```

Il programma del Capitolo 7 contiene un esempio interessante di funzione definita dall'utente.

IL CODICE ASCII

Lo ASCII, l'*American Standard Code for Information Interchange* (codice americano standard per lo scambio delle informazioni), viene utilizzato per trasformare

le lettere, le cifre e i caratteri speciali del BASIC nella rappresentazione numerica con la quale vengono memorizzati nel computer. Ciascun carattere corrisponde a un numero del codice ASCII . La Figura B.4 mostra una tavola ASCII parziale. Tale codice contiene valori da 0 a 255, ma i caratteri rappresentati dalla sua parte alta e da quella bassa tendono a variare a seconda delle versioni.

Il BASIC ha una funzione che trasforma un numero di codice ASCII in un carattere ASCII: la funzione CHR$. Si può lanciare questo breve programma per vedere i caratteri del codice ASCII:

```
10 FOR I = 0 TO 255
20 PRINT CHR$(I)
30 NEXT I
40 END
```

33	!	54	6	75	K
34	"	55	7	76	L
35	#	56	8	77	M
36	$	57	9	78	N
37	%	58	:	79	O
38	&	59	;	80	P
39	'	60	<	81	Q
40	(	61	=	82	R
41	)	62	>	83	S
42	*	63	?	84	T
43	+	64	@	85	U
44	,	65	A	86	V
45	-	66	B	87	W
46	.	67	C	88	X
47	/	68	D	89	Y
48	0	69	E	90	Z
49	1	70	F	91	[
50	2	71	G	92	\
51	3	72	H	93	]
52	4	73	I	94	^
53	5	74	J	95	_

Figura B.4 - Una tavola parziale dei codici ASCII

APPENDICE C

COMANDI SPECIALI DALLA TASTIERA

INTRODUZIONE

I tasti che vanno premuti per spostare il cursore del VisiCalc e per cancellare o inserire i dati dal rigo di edit, variano a seconda dei computer. In questo libro tali comandi sono stati designati con i nomi:

- *Tasti direzionali*, che spostano il cursore, in alto, in basso, a destra o a sinistra.

- *Tasto escape,* che cancella un carattere (dal rigo di edit) tutte le volte che viene premuto.

- *Tasto break*, che cancella l'intero contenuto del rigo di edit, oppure cancella tutto un comando.

- *Tasto return*, che inserisce i dati dal rigo di edit.

I paragrafi che seguono spiegano come impartire tali comandi sui personal computer Apple II; TRS-80, modelli I e III e IBM.

APPLE II

I tasti direzionali. L'Apple II ha due soli tasti direzionali, sulla destra della tastiera, contrassegnati rispettivamente da una freccia verso sinistra e da una verso destra. Questi due tasti consentono di spostare il cursore in tutte e quattro le direzioni: premendo lo spaziatore si stabilisce se devono effettuare uno spostamento verticale o uno orizzontale. Un carattere nell'angolo superiore destro dello schermo comunica la modalità corrente : una lineetta (-) indica lo spostamento orizzontale; un punto esclamativo (!) indica quello verticale. Si noti come questo indicatore cambi tutte le volte che viene premuto lo spaziatore.

Il tasto escape. Questo tasto è marcato ESC ed è nell'angolo in alto a sinistra della tastiera.

Il tasto break. La tastiera dell'Apple II non ha un tasto marcato BREAK. Per eseguire tale comando, vanno premuti due tasti contemporaneamente: quello marcato CTRL e il C. Questa sequenza viene chiamata "control-C".
Il tasto return. Sull'Apple II è appunto marcato RETURN.

TRS-80, MODELLI I E III

I tasti direzionali. Il TRS-80 offre quattro tasti distinti, contrassegnati dalle freccie nelle quattro direzioni.
Il tasto escape. E' quello marcato CLEAR.
Il tasto break. E' marcato BREAK
Il tasto return. E' marcato ENTER.

IL PERSONAL COMPUTER IBM®

I tasti direzionali. Il Personal Computer IBM ha quattro tasti direzionali, disposti nel tastierino numerico.
Il tasto escape. Si utilizza il tasto di arretramento
Il tasto break. E' marcato SCROLL LOCK ed è nell'angolo superiore destro della tastiera.
Il tasto enter. E' marcato ↵.

INDICE ANALITICO

Esperti a confronto su attualità e prospettive della Computer Grafica

Computer Graphics, CAD, elaborazione di immagini: sistemi e applicazioni
A cura di
Alessandro Polistina

Linguaggi e algoritmi, sistemi grafici, CAD/CAM, didattica e formazione professionale, Computer Graphics e Editoria, modellazione di solidi, CAD in architettura, CAD meccanico, acquisizione e elaborazione di immagini, elaborazione di immagini e scienze biomediche, cartografia e pianificazione editoriale, immagini sintetiche per la televisione....

Tutti gli Atti del 3° Convegno Nazionale AICOGRAPHICS riuniti in un solo volume a disposizione di operatori, sperimentatori, appassionati. 512 pagine, numerosissimi schemi, un'Appendice con 33 illustrazioni a colori..
Lire 45.000
Codice **529C**

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	529C	**L. 45.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione

☐ Allego assegno della Banca ____________________
n° ____________________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|

SCONTO 20% AGLI ABBONATI FINO AL 28-2-'84

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

nuovidea

Il "foglio elettronico" è una delle applicazioni più interessanti dei personal computer come supporto alle attività contabili e, soprattutto, alle attività decisionali che dalle prime scaturiscono: i programmi su cui si basa consentono di elaborare simultaneamente centinaia di numeri, esplorare le diverse alternative suggerite da queste elaborazioni e infine trarre le debite conclusioni.
VISICALC è senz'altro uno dei migliori e più diffusi prodotti software di questo tipo, e il suo successo è dovuto ad una estrema semplicità d'uso e alla notevole flessibilità con cui copre una vasta gamma di problemi; non solo, ma con una cordialità rara nel mondo del software, VISICALC è in grado di scambiare dati con altri strumenti software dello stesso ambiente. Questo libro presenta VISICALC e le sue applicazioni, rivolgendosi a diversi livelli di utenza: dal principiante che deve essere guidato "per mano", al conoscitore di VISICALC che desidera ottimizzarne l'uso, all'esperto che intende esplorare le possibilità di utilizzo di VISICALC in combinazione con altri programmi.

L. 15.000 Cod. 556 H ISBN 88-7056-147-X

GRUPPO
EDITORIALE
JACKSON
Douglas Hergert
VisiCalc
92